VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

PORTABILITA DISTRIBUOVANÝCH VÝPOČTŮ V RÁMCI CLOUDOVÝCH INFRASTRUKTUR
PORTABILITY OF DISTRIBUTED COMPUTING IN CLOUD INFRASTRUCTURES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Cuong Tuan Duong

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jiří Kříž, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Cuong Tuan Duong**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Jiří Kříž, Ph.D.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Portabilita distribuovaných výpočtů v rámci cloudových infrastruktur

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Testování přesunu platformy Metapipe pro výpočty metagenomických dat v distribuovaném prostředí mezi privátním cloudem OpenStack a veřejným akademickým cloudem EGI Federated cloud.

Základní literární prameny:

AGAFONOV, A., K. MATTILA a C. D. TUAN. META-pipe cloud setup and execution. 2018. Dostupné z https://doi.org/10.12688/f1000research.13204.2

BRIKMAN, Y. Terraform: Up and Running: Writing Infrastructure as Code. 1. vyd. New York: O'Reilly Media, 2017.

HOCHSTEIN, L. a R. Moser. Ansible: Up and Running, 2. vyd. New York: O'Reilly Media, 2017.

ROBERTSEN, E. M., T. KAHLKE a I. A. RAKNES. META-pipe - Pipeline Annotation, Analysis and Visualization of Marine Metagenomic Sequence Data. ArXiv160404103 Cs. 2016.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně dne 28.2.2019

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

ABSTRAKT

Diplomová práce se zabývá analýzou řešení distribuovaných výpočtů pro metagenomická data v rámci cloudových infrastruktur. Popisuje specifickou platformu META-pipe založenou na architektuře klient-server v infrastruktuře veřejného akademického cloudu EGI Federated Cloud, sponzorovanou evropským projektem ELIXIR-EXCELERATE. Práce se zaměřuje především na open-source software jako Terraform a Ansible.

KLÍČOVÁ SLOVA

distribuované výpočty, open-source, Terraform, Ansible, cloud, EGI, ELIXIR

ABSTRACT

The master's thesis focuses on analysis of solution to distributed computing of metagenomics data in cloud infrastructures. It describes specific META-pipe platform based on client-server architecture in infrastructure of public academic cloud EGI Federated Cloud, sponsored by european project ELIXIR-EXCELERATE. Thesis is focusing especially on open-source software like Terraform and Ansible.

KEYWORDS

distributed computing, open-source, Terraform, Ansible, cloud, EGI, ELIXIR

DUONG, Cuong Tuan. *Portabilita distribuovaných výpočtů v rámci cloudových infrastruktur* [online]. Brno, 2019 [cit. 2019-05-11]. Dostupné z: `https://www.vutbr.cz/studenti/zav-prace/detail/120067`. Diplomová práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Jiří Kříž.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Portabilita distribuovaných výpočtů v rámci cloudových infrastruktur“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jiřímu Kříži, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

ÚVOD **1**

1 CÍL PRÁCE **2**

2 TEORETICKÉ VÝCHODISKA PRÁCE **3**

2.1 Grid computing . . . 3
2.2 Cloud computing . . . 4
2.3 META-pipe . . . 5
2.3.1 Vstupní data . . . 5
2.3.2 Taxonomická klasifikace . . . 6
2.3.3 Assembly . . . 7
2.3.4 Funkcionální analýza . . . 8
2.3.5 Vizualizace výsledků analýzy . . . 9

3 CLOUDOVÉ INFRASTRUKTURY V EU **10**

3.1 ELIXIR . . . 10
3.1.1 ELIXIR-EXCELERATE . . . 12
3.2 EGI . . . 14
3.2.1 EGI-Engage . . . 15
3.3 EGI Federated Cloud . . . 16
3.3.1 Architektura . . . 17
3.3.2 Specifické kapacity cloudu . . . 18
3.3.3 Integrace do EGI Federated Cloud . . . 19
3.3.4 Vývoj EGI Federated Cloud . . . 20
3.3.5 Podpora uživatelů . . . 21
3.4 NGI . . . 23
3.4.1 MetaCentrum . . . 23

4 ANALÝZA SOUČASNÉHO STAVU **25**

4.1 Implementace a integrace META-pipe v NeLS . . . 25
4.1.1 Používání META-pipe . . . 25
4.1.2 Galaxy . . . 26
4.1.3 Integrace úložiště . . . 27
4.1.4 Integrace superpočítače . . . 28
4.2 Obecná integrace META-pipe a běh klienta . . . 29
4.2.1 Server . . . 29
4.2.2 Klient . . . 30
4.2.3 MMG Cluster setup . . . 31

5 NÁVRH ŘEŠENÍ **32**
5.1 Open Cloud Computing Interface . . . 32
5.1.1 rOCCI-cli . . . 33
5.2 Terraform . . . 34
5.2.1 Porovnání s jinými nástroji . . . 35
5.3 Ansible . . . 36
5.3.1 Architektura . . . 36
5.4 Python . . . 38
5.5 MMG Cluster Setup - CESNET . . . 39

6 IMPLEMENTACE ŘEŠENÍ **41**
6.1 Terraform OCCI modul . . . 41
6.1.1 Kontextualizace . . . 42
6.1.2 Create . . . 43
6.1.3 Read . . . 45
6.1.4 Update . . . 46
6.1.5 Delete . . . 46
6.2 Ansible playbooks . . . 46
6.2.1 Role . . . 47
6.3 MMG Cluster Setup . . . 50
6.3.1 create.py . . . 50
6.3.2 run.py . . . 52
6.3.3 stop.py . . . 52
6.3.4 destroy.py . . . 52

7 OPTIMALIZACE **53**
7.1 CernVM File System . . . 53
7.2 Squid . . . 56
7.3 Úprava návrhu . . . 58
7.3.1 Ansible role . . . 58
7.3.2 Upravená architektura . . . 59
7.4 Benchmarking . . . 60
7.5 Docker . . . 61
7.5.1 Komponenty . . . 62
7.5.2 Nástroje . . . 64
7.5.3 Použití v META-pipe . . . 64

8 ZÁVĚR **66**

Literatura **67**

ÚVOD

Cloud je v dnešní době fenomén. Takřka každý o něm slyšel, od úplných laiků po vrcholné manažery. Principiálně se ale jedná o to, co se využívá již desetiletí - použití cizích počítačů na síti k výpočtům, také zvané jako distribuované výpočty.

Pod pojmem distribuované výpočty se myslí většinou zpracování úloh, které lze snadno paralelizovat a distribuovat jejich části jiným počítačům. Pro to lze použít cokoliv od Raspberry Pi[1] přes počítačové učebny po obří datacentra vlastněná státy či korporacemi jako Google nebo Facebook.

Distribuované výpočty mají velkou historii - jedna z prvních distribuovaných aplikací byl email v ARPANETu[2]. Pro tyto účely byly vytvořená specializovaná datacentra, nicméně v poslední době se od toho pomale ustupuje a používají se především infrastruktury na bázi Cloud computingu. Ty se liší tím, že jsou vysoce škálovatelné a nabízí možnost běhu vlastního virtualizovaného systému.

Jedna z aplikací využívající právě distribuovaných výpočtů na bázi cloud computingu je META-pipe vyvinutá Univerzitou v Tromsu. Zpracovává biologická data, která jsou nesmírně náročná na zpracování, ale výpočetně jsou úlohy snadno paralelizovatelné, proto se využívá distribuovaných výpočtů.

[1] Malý a levný počítač určený především k výuce informatiky

[2] Advanced Research Projects Agency Network, technologický předchůdce internetu vyvinut Ministerstvem obrany USA

1 CÍL PRÁCE

Cílem práce je zaměřit se na současnou implementaci služby META-pipe, její fungování a možnost rozšíření jejího fungování do jiného distribuovaného prostředí. V rámci spolupráce sdružení CESNET a EGI bylo zvoleno prostředí EGI Federated Cloud.

V první části se práce zaměří na teoretická východiska práce vycházející z heuristické analýzy expertů v oboru. Součástí práce by měl být i popis institucí zapojených do provozu této služby a prostředí, na kterých služba poběží. Následně práce bude obsahovat analýzu současného stavu, popis jednotlivých součástí současné implementace a integrace META-pipe. Následně se práce zaměří na samotnou implementaci nástroje, kterým se bude integrovat META-pipe do EGI Federated cloud. V poslední kapitole se rozvine i optimalizace implementace vycházející z úpravy architektury původního návrhu META-pipe.

2 TEORETICKÉ VÝCHODISKA PRÁCE

V této kapitole se bude práce zabývat technologiemi, které se využívají v distribuovaném prostředí a jak se využívají. Následně se bude práce zabývat popisem služby META-pipe, funkcemi, kterými disponuje a jak fungují.

2.1 Grid computing

Grid computing je způsob využití distribuovaných prostředků k dosažení cíle. Výpočetní grid se může brát jako distrubovaný systém s neinteraktivní[1] pracovní zátěží využívající velké množství souborů. Grid computing se liší od běžných HPC[2] tím, že každý uzel v gridu je přiřazen na jinou úlohu či aplikaci. Grid tím pádem je typicky více heterogenní a geograficky rozprostřený - např. grid provozovaný sdružením CESNET se nachází v lokalitách po celé České republice, zejména Praze a Brně. Ačkoliv jeden grid může být dedikovaný jedné aplikaci, typicky je využit pro vícero aplikací.

Grid je forma distrubovaných výpočtů, kde jeden virtuální "superpočítač"je složen z mnoha počítačů, které jsou připojené sítí, obvykle internetem, které spolu spolupracují na úkolech. Tím způsobem se dá říct, že grid computing je forma paralelních výpočtů, která závisí na kompletních počítačech (narozdíl třeba od jednotlivých jádrech CPU) připojených k síti. To je velký rozdíl od tradičních superpočítačů, který má velké množství CPU nebo GPU připojené přes lokální vysokopropustnou síť, např. InfiniBand[3].

Gridy mohou být jak soukromé (např. pro komplikované výpočty umělé inteligence v Google), tak veřejné pro účely náročných výpočtů v oborech matematiky, chemie apod. Takové gridy mohou dosahovat velikosti tisíce počítačů, např. české MetaCentrum na sklonku roku 2017 nabízelo více jak 15 000 CPU jader společně s rozsáhlými úložnými kapacitami přesahující 4 PB.

Grid má velkou výhodu v tom, že je principiálně tvořen heterogenními počítači. To má využití v tom, že do některých veřejných gridů se může zapojit kdokoliv se svým počítačem. Toho využívá např. projekt SETI@home Kalifornské Univerzity v Berkeley, který vyhledává známky mimozemské civilizace. Existují i podobné projekty s jiným zaměřením, např. Einstein@Home.

[1] Nevyžadující zásah uživatele

[2] High performance computing - clustery, superpočítače

[3] Vysokopropustný komunikační standard s nízkou latencí. Zdaleka nejpoužívanější připojení v rámci superpočítačů

2.2 Cloud computing

Cloud computing je on-demand dostupnost výpočetních prostředků, především úložišť a výpočetního výkonu bez aktivního řízení uživatelem. Tento pojem je obecně používán k popisu datacenter dostupných uživatelům po celém světě skrze Internet. Velké cloudy jsou často distribuované po celém světě.

Cloudy mohou být jak privátní enterprise cloudy, tak veřejné či hybridní. Aktuálně největším veřejným cloudem je Amazon AWS.

Cloud computing je stavěný na principu sdílení prostředků mnoha uživateli. Tím způsobem je dosažena nízká cena a vysoká dostupnost. Proto je často cloud využíván menšími organizacemi, kde není potřeba řešit vlastní datacentrum, ale pronajímá se tak virtuální datacentrum v cloudu. Cloudy jsou obecně zpoplatněné systémem "pay as you go", kde se neplatí předplatné, ale platí se za skutečně využitou hodnotu (typicky se počítá v CPU sekundách), obvykle měsíčně.

Mezi typické charakteristiky cloudu patří:

- Multitenancy - Na jednom zdroji je více uživatelů, kteří ho sdílejí
- Škálovatelnost a elasticita - cloud je díky on demand modelu principiálně velmi flexibilní a nabízí možnost škálování za běhu, např. zvýšení počet jader CPU při zvýšeném vytížení
- Pay as you go - Placení za spotřebovaný výkonu
- Jednoduchá či nulová údržba - Údržbu a aktualizace systému za uživatele dělá provozovatel cloudu
- Bezpečnost - Na bezpečnosti cloudu se podílí tým expertů provozovatele cloudu, kteří udržují aktualizovaný systém

Mezi distribuční modely cloudu patří:

- IaaS - Infrastruktura jako služba, kdy se nabízí funkcionalita systémové integrace na úrovni virtuálních strojů běžících nad hypervizorem, např. Xen, KVM, VMware ESX/ESXi apod.
- PaaS - Platforma jako služba, kdy se nabízí jistá softwarová platforma pro běh aplikací, např. Operační systémy, middleware apod. Typicky PaaS běží na IaaS
- SaaS - Software jako služba, kdy se nabízí přímo aplikace, hostovaná na cloudu poskytovatele. Většina webových aplikací se dá považovat za SaaS, mezi ně patří například Google Apps, iCloud, Microsoft Office apod.

Cloud je momentálně velmi populární díky své flexibilitě, nicméně velké množství kritiků, mezi nimi i zakladatel projektu GNU Richard Stallman poukazují na nebezpečí ukládání dat do cloudu a tím pádem ztrátou soukromí uživatelů.

2.3 META-pipe

META-pipe je ucelená služba pro analýzu mořských metagenomických dat vyvinutá v Univerzitě v Tromsu. V současnosti funguje na bázi architektury klient-server, kdy klient, tzv. "execution manager", získává ze serveru data, která má dále zpracovat. Existují 3 centrální servery a několik geograficky distribuovaných execution managerů, mezi které patří:

- Superpočítač Sigma2 Stallo v Tromsu[4]
- CSC cPouta[5]
- Amazon EMR

Název vychází z procesu, který uceluje - METAgenomic pipeline[6]. Nabízí zpracování sekvenčních dat s vysokou propustností a funkcionální anotaci[7] predikovaných genů. Funkcionální anotace je velmi výpočetně náročná a nyní běží na high-performance výpočetním clusteru[8] v Norsku. Nicméně pro nabídnutí služby širšímu okruhu uživatelů je potřeba většího výpočetního výkonu, než je dostupné.

V momentální iteraci META-pipe nabízí taxonomickou a funkcionální analýzu pro 16S amplikon[9] dat a data tzv. whole genome shotgun sequence[10]. Podporuje sekvenční data s vysokou propustností a poskytuje sequence assembly[11]. Pipeline se skládá ze tří hlavních modulů: pre-processingu, taxonomické klasifikace a funkcionální analýzy. Jednotlivé moduly jsou dostupné jako individuální workflows kromě assembly v pre-processingu, která je zpracována manuálně ve výpočetním clusteru nebo superpočítači. Každá workflow může být specificky upravena pro potřeby analýzy vzorku.

2.3.1 Vstupní data

META-pipe akceptuje surová sekvenční data jako prvotní výstup. Před analýzou META-pipe pre-processuje surová data a vyfiltruje data nízké kvality s pomocí PRINSEQ[12], buď s manuálně zvolenými parametry nebo parametry, které jsou spe-

[4]High performance cluster pro Norskou akademickou obec

[5]Součást CSC - Finské superpočítačové centrum pro vědu

[6]V výpočetní technice je pipeline série elementů zpracovávající data, kde výstup jednoho je vstup druhého - např. grafická nebo instrukční pipeline

[7]Proces sbírání informací o identitě a funkci jednotlivých genů. Jedná se o proces následující sekvenování genomu

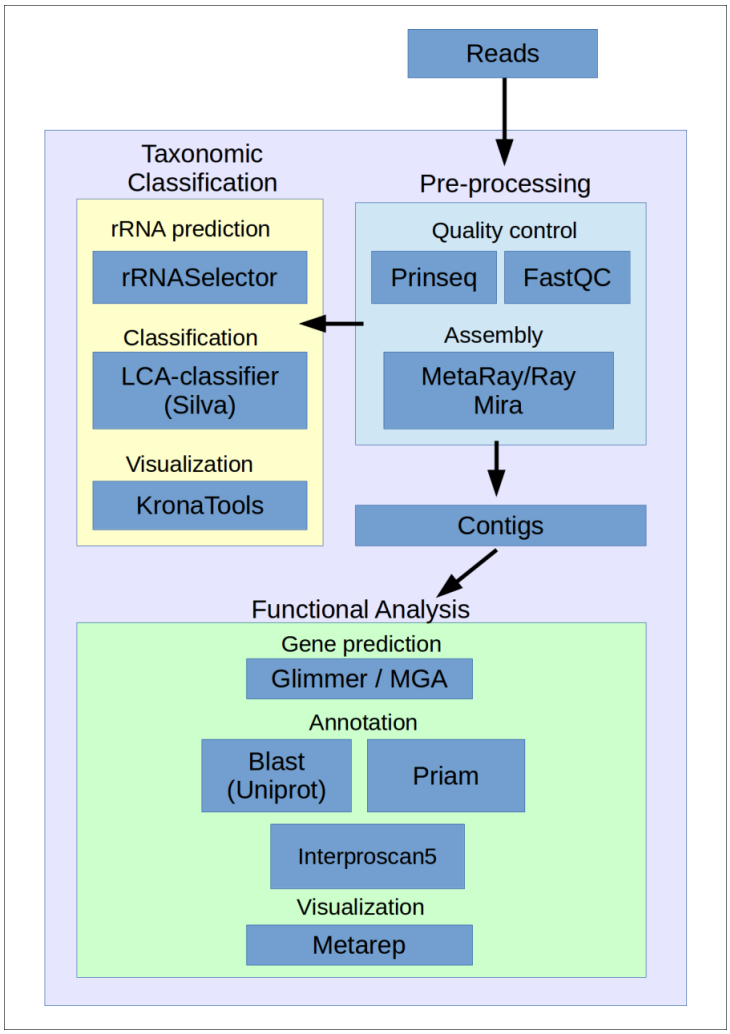
[8]Počítačový cluster je seskupení počítačů, které spolu úzce spolupracující

[9]Kus DNA nebo RNA, která je zdrojem nebo produktem amplifikace či replikace

[10]tzv. shotgun sequencing je metoda používaná pro sekvenování náhodných DNA řetězců

[11]Proces, při kterém se sbírají fragmenty DNA a skládají se dohromady pro rekonstrukci původních chromozomů ze kterých DNA pochází

[12]Open-source nástroj pro filtraci, formátování a ořezávání sekvenčních dat široce využívaný v bioinformatické komunitě



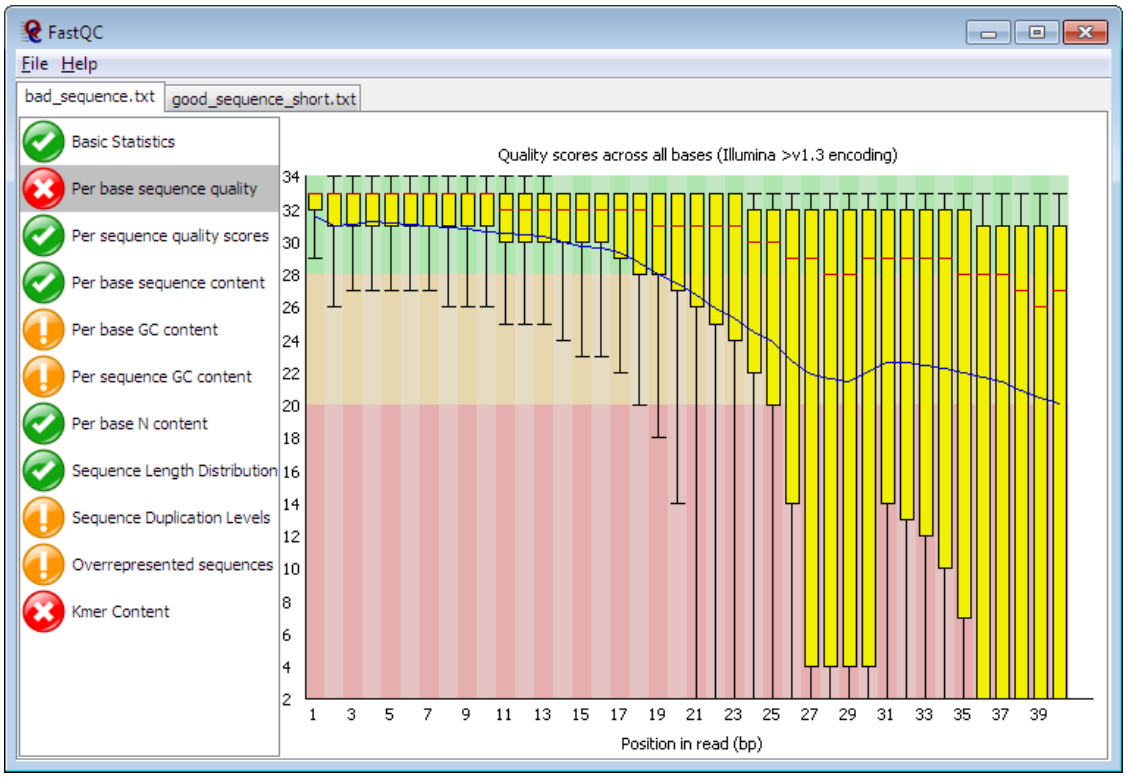
Obr. 2.1: Schéma jednotlivých komponentů a modulů META-pipe [1]

ciálně upravené pro užitou sekvenční platformu jako Illumina[13]. Uživatel si může zkontrolovat výsledky filtrovacího procesu pomocí grafů z FastQC[14].

2.3.2 Taxonomická klasifikace

Modul taxonomické klasifikace používá filtrované sequence k vytvoření přehledu organismů přítomných v datovém souboru. První krok ve workflow modulu je použití rRNA[15] selektoru k predikci a ořezání rRNA sekvencí ze čtení. Predikované sekvence se porovnávají s vybranou verzí SILVA databáze[16] a následně vstupují do klasifikač-

[13] Americká společnost specializující se na analýzu genetických variací a biologických funkcí
[14] Nástroj pro kontrolu kvality surových sekvenčních dat
[15] Ribozomální RNA je druh RNA podílející se na tvorbě ribozomu
[16] Databáze rRNA provozovaná Institutem Maxe Plancka v Brémách



Obr. 2.2: Program FastQC [2]

ního programu LCA-classifier. Ten používá algoritmus "lowest common ancestor"[17] pro klasifikaci do jednotlivých taxonomických jednotek které jsou následně zobrazeny v interaktivním grafu programu Krona.

Uživatelé META-pipe můžou použít modul taxonomické klasifikace taky k maskování predikovaných 16S rRNA sekvencí z datového souboru, vylučující je z kroku assembly a tudíž redukování nepravých contigů[18] v assembly. Modul taktéž umí klasifikovat 16S amplikon datové soubory.

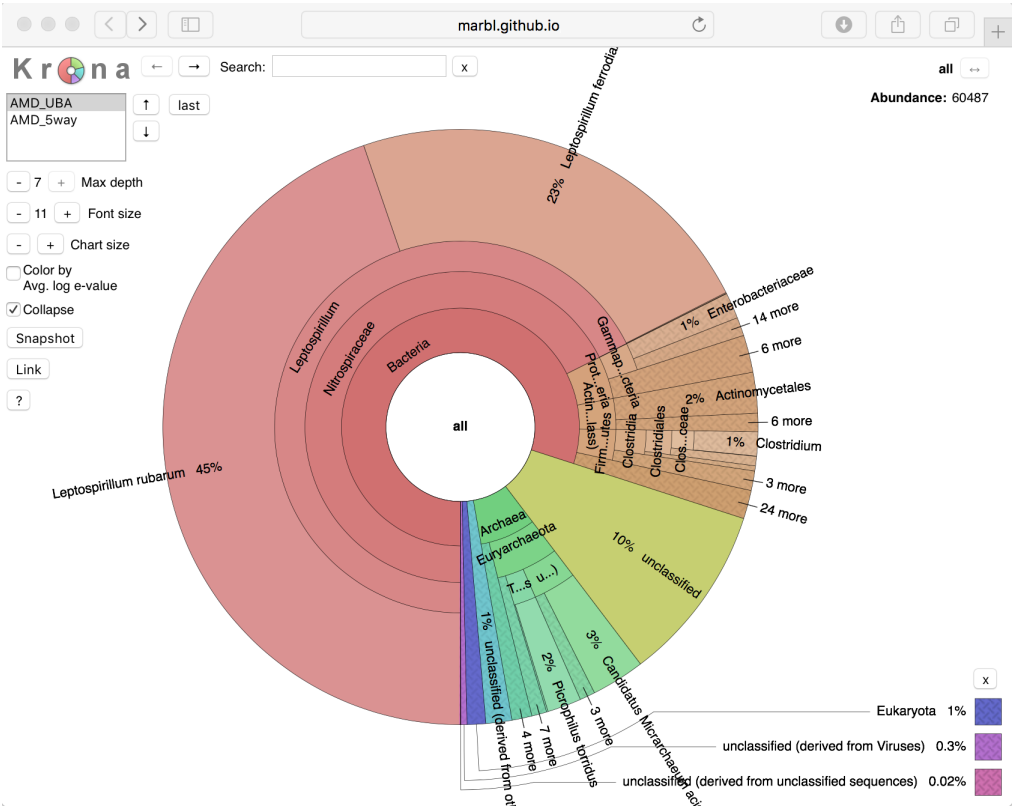
2.3.3 Assembly

Assembly je krok, kdy se v META-pipe skládají datové soubory k vzniku souboru contigů pro fukcionální analýzu. Skládání metagenomických datových souborů typicky vyžaduje nemalé množství operační paměti kvůli komplexitě a hloubce sekvenování. Množství použité operační paměti může dosahovat stovek GB, závisející na použitém assembly programu a na velikosti a typu vstupního datového souboru. Pro překonání tohohle problému META-pipe nabízí dva různé assembly programy:

- Mira - pro genomy a menší sekvenční datové soubory běžící na jednom uzlu

[17] Algoritmus pro vybrání nejbližšího společného předka ve stromu nebo orientovaném acyklickém grafu

[18] Delší segmenty DNA ze vzorku



Obr. 2.3: Zobrazení hiearchických dat v programu Krona [3]

- Ray/MetaRay - pro větší metagenomické sekvence kde se výpočet a datové struktury distribuuje mezi více uzlů a komunikace mezi uzly je implementovaná pomocí MPI[19]

Tyto programy se automaticky vybírají podle velikosti vstupního datového souboru, nicméně je možné manuálně zvolit jednotlivý program.

2.3.4 Funkcionální analýza

Vstupem modulu fukcionální analýzy jsou contigy vytvořené v kroku assembly. Tento modul je založen na GePan anotační pipeline pro sekvenční data genomů. V GePan byla rozšířena funkcionalita a škálovatelnost pro podporu analýzu metagenomických datových souborů.

Prvním krokem modulu funkcionální analýzy je predikce sady genů z poskytnutého vstupu contig sekvencí pomocí GLIMMER[20] nebo MGA[21]. Tato sada je

[19]Message Passing Interface je knihovna implementující protokol pro paralelní řešení výpočetních problémů v clusterech

[20]Gene Locator and Interpolated Markov ModelER je program na hledání genů v prokaryotické DNA

[21]MetaGeneAnnotator je program pro hledání genů fágů a prokaryotů vyvinut Národním insti-

rozdělena relativně k počtu využitých CPU jader pro efektivní škálovatelnost s dostupnými prostředky. Druhým krokem je běh vybraných dostupných nástrojů na každém datovém souboru rozdělených paralelně. META-pipe nabízí následující nástroje:

- Blast s databází UniprotKB[22]
- Priam pro identifikaci enzymů
- Interpro nabízející dalších 11 databází

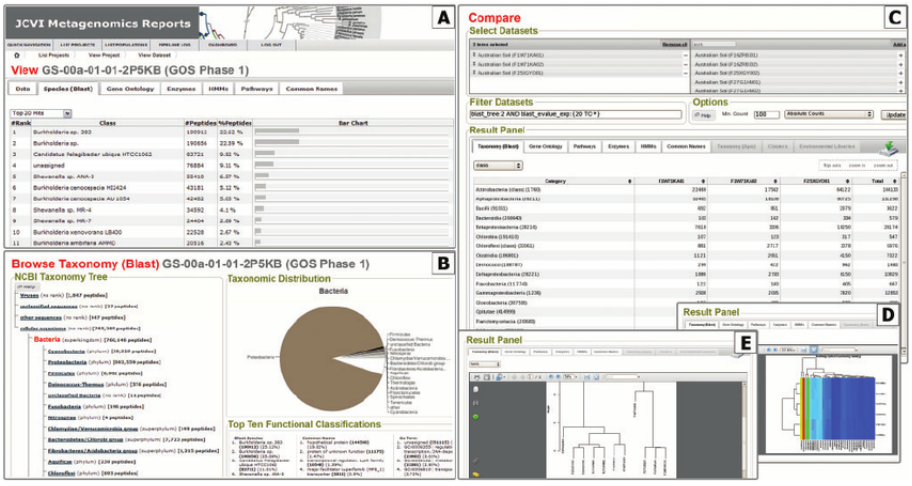
Každý z nástrojů produkuje sadu výstupů která jsou zpracována anotačním modulem, produkujícím soubor spojující predikované geny s jejich funkcionální analýzou.

2.3.5 Vizualizace výsledků analýzy

Anotační modul exportuje výstupní data pro vizualici pomocí samostatných programů. META-pipe nabízí tyto formáty:

- XML
- EMBL
- Text oddělený tabulátory
- Formát METAREP visualization tool

METAREP je open-source nástroj s webovým interface pro high-performance komparativní metagenomickou vizualizaci. Umožňuje zobrazení, prohlížení a porovnávání datových souborů pomocí věstavených nástrojů jako jsou statistické testy, multidimenzionální heatmaps a analýzy cesty. META-pipe obsahuje modifikovanou verzi, která navíc nabízí znovuzískání sekvence, což umožňuje stáhnutí a další vyšetření konkrétních genů.



Obr. 2.4: Webový interface programu METAREP [4]

tutem genetiky v Japonsku

[22]UniProt Knowledgebase je středisko pro sběr funkcionálních informací o proteinech

3 CLOUDOVÉ INFRASTRUKTURY V EU

Do distribuovaných výpočtů a cloudu je zapojená řada organizací v Evropě, některé nabízí univerzální výpočetní prostředí jako např. EGI, některé nabízí infrastrukturu specificky pro užití v jednom oboru, např. ELIXIR pro life sciences[1].

3.1 ELIXIR

ELIXIR je evropská mezinárodní organizace pro spolupráci v oboru life sciences. Má 21 členských států a více jak 180 spolupracujících výzkumných organizací. Byla založená v roce 2014 a právě implementuje svůj první pětiletý výzkumný program.

Mezi jednotlivými členy existuje úzká spolupráce. Například ELIXIR Norway nabízí právě službu META-pipe, pro kterou ELIXIR Finland nabízí vhodné výpočetní prostředí.

Mise ELIXIRu je budování dlouhodobě udržitelné Evropské infrastruktury pro biologické informace, podporu life sciences výzkumu a jeho využití v medicíně, bioprůmyslu a společnosti.

Výsledky z biologických experimentů produkují velké množství výsledků které jsou ukládány jako počítačové data. Evropské země investovaly nemalé částky do výzkumů, které produkují, analyzují a skladují biologické informace. Nicméně sbírání, skladování, archivace a integrace těchto dat způsobuje problémy, které jsou neřešitelné pro jednu zemi. ELIXIR reprezentuje spolupráci zařízení v jednotlivých zemích pro integrovanou síť, která je uzpůsobená k řešení komplexního problému skladování biologických dat. Tato spolupráce vyúsťuje v tvorbě jednotné infrastruktury, která zjednodušuje vědcům hledání a sdílení dat, výměnu odborných názorů a dohody best practices.

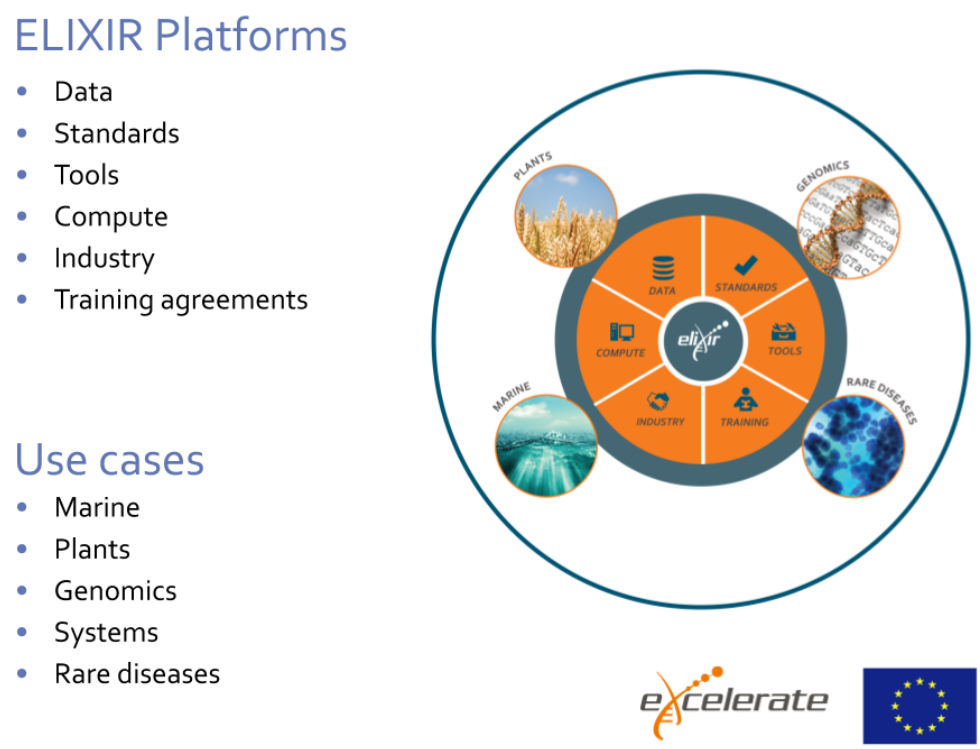
Jedna z platforem ELIXIRu je TeSS - výuková platforma. Jedná se o online výukový portál, který sbírá výukové materiály v oblasti life sciences v rámci Evropy a umožňuje jejich vyhledávání na jedné webové stránce. To zjednodušuje vědcům získat trénink v oblastech, kde se chtějí rozvinout a zároveň zvyšuje povědomí o výukových kurzech, které tak mohou být dostupnější širší veřejnosti.

ELIXIR se mimo jiné zaměřuje především tyto platformy:

- Data
- Standardy
- Nástroje
- Výpočty

[1]Dá se přeložit jako přírodní vědy zahrnující nejen čistou biologii, ale třeba bioinformatiku, immunologii apod.

- Průmysl
- Tréninkové dohody



Obr. 3.1: Platforma ELIXIR [5]

3.1.1 ELIXIR-EXCELERATE

ELIXIR-EXCELERATE je projekt organizace ELIXIR zařazený do Horizon 2020[2]. Jedná se o čtyřletý projekt s rozpočtem 19 milionů € v koordinaci 17 států a 41 partnerů. Účelem projektu je akcelerace implementace ELIXIR infrastruktury u jednotlivých členů. To zahrnuje budování bioinformatické výpočetní kapacity v rámci Evropy.

Čtyři use cases, které mají využívat tyto kapacity jsou tyto:

- Rare diseases - vzácné nemoci
- Human data - lidská data
- Plant genotype/phenotype - genotyp/fenotyp rostlin
- Marine metagenomics - Mořská metagenomika

Projekt se dělí na několik Work Packages, kde každá se zaměřuje na jednu z platforem či use case. Platformy a Use cases jsou navzájem provázané.

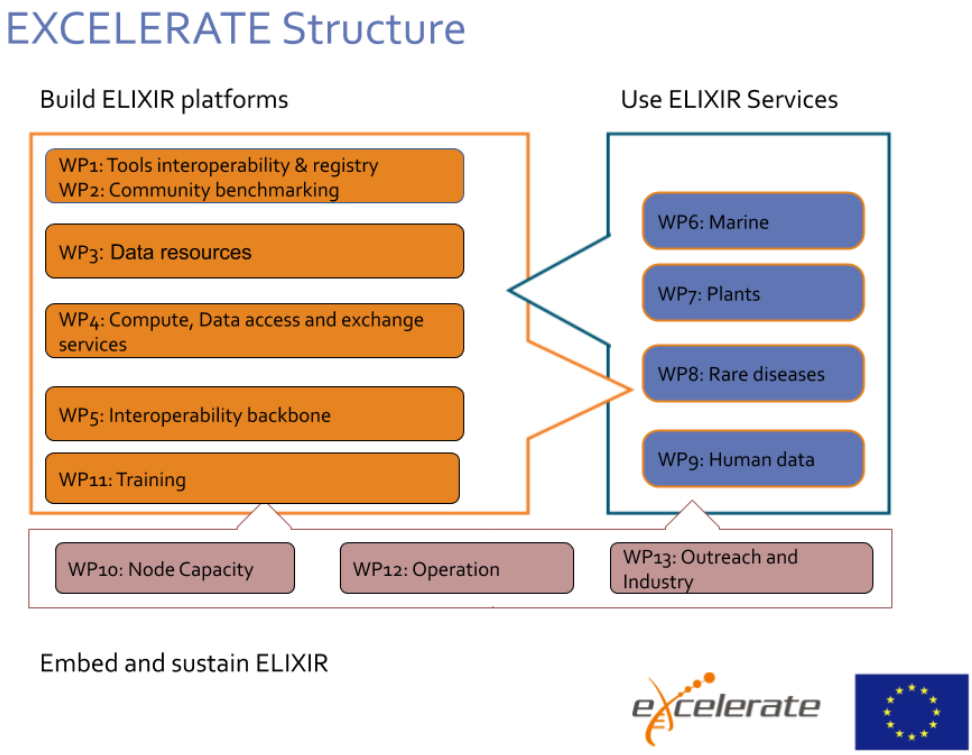
Mezi work packages patří:

- WP1: Tools Interoperability and Service Registry - registr interoperabilit nástrojů a služeb
- WP2: Infrastructure for Community-led Benchmarking - Zhodnocení bioinformatických metod v kvantitativním měřítku a uživatelské přívětivosti
- WP3: Data Resources and Services - Budování frameworku pro vývoj datových prostředků a služeb
- WP4: Compute, Data access and Exchange Services - Výpočetní služby
- WP5: The ELIXIR Interoperability Backbone - Interoperabilita nástrojů a služeb v ELIXIR
- WP6: Marine Metagenomic Infrastructure as Driver for Research and Industrial Innovation - Výzkum mořské metagenomiky
- WP7: Integrating Genomic and Phenotypic Data for Crop and Forest Plants - integrace genomu/fenotypu pro rostliny
- WP8: ELIXIR Infrastructure for Rare Disease Research - Infrastruktura pro výzkum vzácných nemocí
- WP9: Secure Archiving, Dissemination and Analysis of Human Access-controlled Data - Šifrování a ochrana citlivých biomedicínských dat
- WP10: ELIXIR Node Capacity Building Programme - Program pro nové členy ELIXIRu
- WP11: ELIXIR Training Programme - Trénovací program pro vědce v rámci ELIXIR

Obecně problém life science výzkumu v Evropě je to, že je zde velké množství vědců, kteří tvoří nezanedbatelné množství dat, které jsou neorganizované a nekom-

[2]Rámcový program Evropské Unie pro výzkum a inovaci v období 2014-2020

patibilní s daty jiných vědců. Tento problém se snaží řešit vytvořením sady nástrojů a infrastruktury, která by byla využívaná evropskými bioinformatiky. V rámci tohoto projektu se za Českou Republiku angažuje akademické sdružení CESNET, kde ve Work Package 4 vede profesor Luděk Matyska vývoj vědecké platformy pro distribuovanou infrastrukturu pro Cloud, úložné prostory a autentizaci. Tento Work Package je silně provázaný mimo jiné s Work Package 6, kde se řeší Marine Metagenomics, pod kterou spadá právě služba META-pipe.



Obr. 3.2: Struktura EXCELERATE projektu [5]

3.2 EGI

European Grid Infrastructure je federovaná e-infrastruktura zřízená k poskytování pokročilých výpočetných služeb pro výzkum a inovaci.

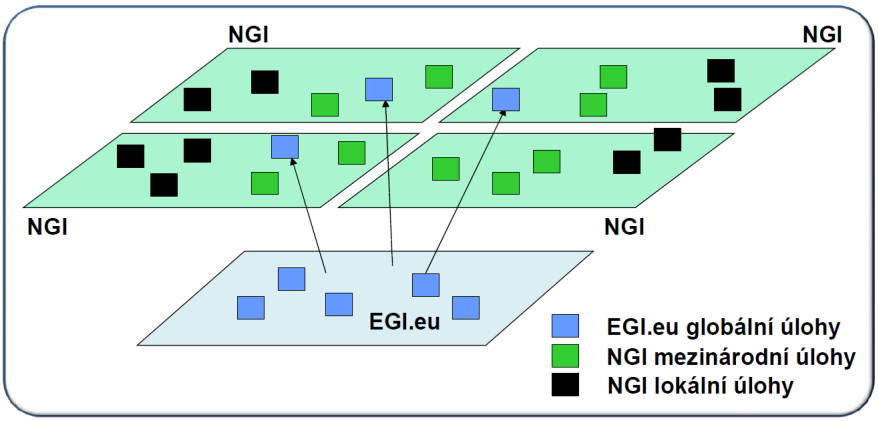
EGI vychází z projektu European DataGrid založeného v roce 2001 a jeho pokračovatelem EGEE (Enabling Grids for E-Science in Europe), vedené divizí informačních technologií CERNu[3]. Ke konci projektu EGEE začala vznikat iniciativa EGI, zaměřená především na National Grid Infrastructures (NGIs).

EGI e-infrastuktura zastřešuje národní NGI, je veřejně financována a skládá se ze stovek data center a cloudových poskytovatelů po celé Evropě. Do EGI jsou taky zapojené některé subjekty mimo Evropu.

Kategorie úloh běžící v NGI/EGI:

- Globální úlohy
  - Úlohy nezbytné pro celkovou koordinaci celé evropské gridové infrastruktury
  - Plná zodpovědnost leží na EGI
  - Provoz infrastruktury a její bezpečnost
  - Interakce s uživateli a jejich podpora včetně zajištění dostupnosti a použitelnosti aplikačního programového vybavení
- Mezinárodní úlohy
  - Úlohy, které musí zajišťovat každá NGI, aby vznikla skutečně propojená jednotná infrastruktura, vychází ze sdílené zodpovědnosti za mezinárodní spolupráci, propojují mezinárodní a národní úroveň gridové infrastruktury
  - Úlohy, které jsou ve společném zájmu dvou a více zemí (NGI, resp. uživatelských komunit)
  - Různá velikost NGI vede přirozeně k různému rozsahu zapojení do mezinárodní spolupráce
- Lokální úlohy
  - Vše ostatní, co nepřesahuje zájem či potřeby jedné země

[3]European Organization for Nuclear Research, organizace provozující největší urychlovač částic na světě - LHC



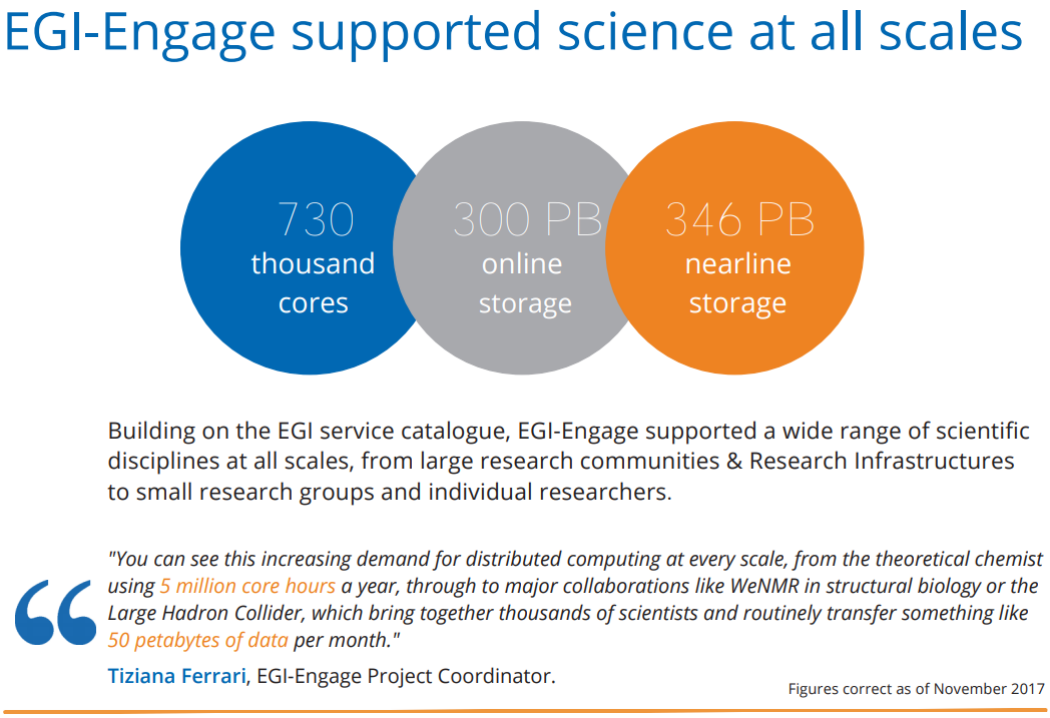
Obr. 3.3: Interakce mezi EGI a NGI [6]

3.2.1 EGI-Engage

Mise projektu EGI-Engage je akcelerace implementace vize Open Science Commons, kde vědci ze všech odvětví mají jednoduchý a otevřený přístup k inovativním digitálním službám, datům a vědomostem, které potřebují k výzkumu. Open Science Commons je založená na třech pilířích:

- e-Infrastructure Commons - ekosystém klíčových služeb
- Open Data Commons - přístupná databáze pro ukládání dat
- Knowledge Commons - sdílená knihovna vědomostí a znalostí pro komunity vědců ke spolupráci vývoje software

EGI-Engage rozšiřuje kapacity dostupné vědcům (např. vylepšený cloud nebo datové služby) tím, že se zapojí do projektů s jinými velkými výzkumnými organizacemi jako je ELIXIR. Hlavní nástroj zapojení je síť osmi tzv. Competence center, kde NGI, uživatelské komunity a poskytovatelé služeb se společně zapojí do sbírání požadavků, integrací specifických aplikací pro jednotlivé komunity a interoperability jednotlivých e-Infrastruktur přes. Tento projekt taky koordinuje úsilí NGI pro podporu služeb a zdrojů, která snižují bariéru pro výzkum a vývoj. V rámci projektu EGI-Engage byl taky navržen EGI Federated Cloud.



Obr. 3.4: EGI-Engage v číslech [7]

3.3 EGI Federated Cloud

EGI Federated Cloud je otevřený cloudový systém, který nabízí škálovatelnou a flexibilní e-infrastrukturu pro Evropskou vědeckou komunitu a rozvíjí EGI výpočetní kapacity za hranici tradičního High Throughput Computing gridové platformy novými modely jako jsou například dlouho běžící služby a výpočty on demand. EGI Cloudové technologie umožňuje federaci institučních cloudů k běhu vědecých výpočtů, simulací a služeb pokrývající několik administrativních lokací. To umožňuje uživatelům přístup a využití takových zdrojů jako unikátního systému. Architektura federace byla definována po dvou letech vývoje založená na sadě uživatelských potřeb popisujících operace na cloudových infrastrukturách a oficiálně byla spuštěna v květnu 2014. Od té doby EGI Federated Cloud operuje jako federace heterogenních IaaS[4] cloudů, kde každý poskytovatel infrastruktury implementoval stejnou sadu rozhrání pro uživatele a systémové administrátory. Prosazení agnosticimus cloudových technologií a podpora mobility služeb pomocí otevřených standardů také umožňuje inkluzi komerčních cloudových poskytovatelů do infrastruktury dříve podporovanou pouze akademickými institucemi. To přispívá k širšímu cílu vytvořit ekonomický a sociální dopad z podpořených vědeckých aktivit.

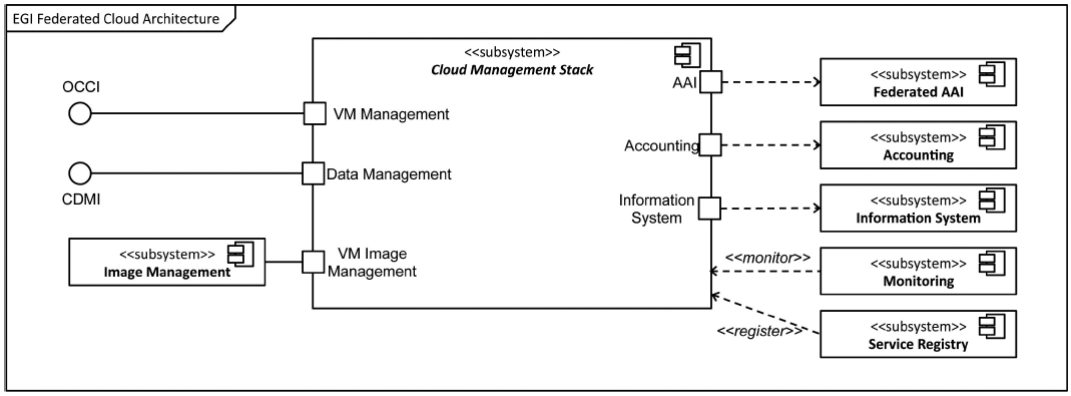
[4]Infrastructure as a Service - platforma, která nabízí infrastrukturu, např. virtuální stroje jako službu

3.3.1 Architektura

Architektura EGI Federated Cloud je výsledek analýzy vysokoúrovňových kapacit získaných z uživatelských požadavků beroucí v potaz i potřeby a odbornost existujících heterogenních cloud management software lokálně instalovaných u poskytovatelů prostředků EGI. Tato výzva je řešená uvažováním každého lokálního cloudu jako autonomním a abstraktním podsystémem, který se integruje do federace přes vhodně navržené rozhrání. Otevřené systémy jsou použity kdekoliv je to možné pro tyto rozhrání pro omezení tzv. vendor lock-in[5]. Každý poskytovatel prostředků by měl identifikovat a nasadit takové řešení, které slouží specificky jejich individuálním požadavkům, ale zároveň musí zajistit podporu vyžadovaných rozhrání.

Architektura definuje specifické kapacity cloudu:

- Virtual Machine (VM) management
- Block storage[6] management
- Data management
- Image management



Obr. 3.5: Architektura EGI Federated Cloud [8]

V současné době EGI Federated Cloud integruje prostředky z několika platforem pro management cloudového prostředí:

- OpenNebula
- OpenStack
- Synnefo

[5]Proprietární uzamčení činí uživatele závislým na produktech a službách konkrétního subjektu, např. pomocí nestandardizovaných součástek

[6]Blokové úložiště je definováno jako sekvence bajtů či bitů s maximální délkou zvanou blok

3.3.2 Specifické kapacity cloudu

V předchozí kapitole zmíněně specifické kapacity cloudu jsou závazné pro poskytovatele prostředků. Poskytovatelé můžou nabídnout VM management, Data management nebo oboje. Pokud je poskytován VM management, je povinná i integrace Image managementu.

VM management

V rámci VM managementu je využívaný RESTový protokol Open Cloud Computing Interface, kterému se práce bude věnovat v návrhové kapitole. Pro EGI Federated Cloud OCCI nabízí interoperabilitní přístup k prostředkům různých poskytovatelů v heterogenním prostředí. Poskytovatelé prostředků v EGI Federated cloudu implementují infrastrukturní specifikace OCCI. To je rozšířené EGI podporou kontextualizace[7] a použitím konzistentní nomenklatury pro tzv. templaty prostředků, též známé pod pojmem "flavours".

Kontextualizace je podporována užitím standardu cloud-init a rozšířením OCCI speficiké pro EGI Federated Cloud, které dovoluje uživatelům definovat data, která mají být předána standardu cloud-init pro kontextualizaci.

OCCI implementace jsou poskytovány jako komponenty pro dříve zmíněné platformy pro management cloudového prostředí:

- rOCCI pro OpenNebula. rOCCI se dá použít nejenom pro platformu OpenNebula, ale i pro CloudStack a Amazon AWS.
- OCCI-OS pro OpenStack. Tento plugin pro OpenStack byl již vyvinut dříve a byl převzat jako OCCI implementace členy komunity EGI
- snf-occi pro Synnefo

Data management

Data management je poskytován jako objektové úložiště[8] s podporou Cloud Data Management Interface(CDMI). CDMI definuje REST rozhrání pro operaci s objekty úložiště. Nabízí klientům způsob jak operovat jak se storage management system (datové kontejnery), tak s jednotlivými datovými jednotkami uvnitř datového kontejneru. Každá implementace CDMI může nabízet rozdílnou sadu možností, které uživatelé zjišťují jako část protokolu.

[7] V prostředí virtualizace kontextualizace dovoluje po instalaci virtuálního stroje přepisování přednastavených hodnot. Užívá se pro prvotní konfiguraci

[8] Typ úložiště, které se od blokového liší tím, že nemá pevnou velikost a využívá tzv. objekty. Objekty se skládají z flexibilně velkých dat a metadat popisující objekt

V prostředí EGI Federated Cloud je referenční implementace CDMI vyvinutá platformou Synnefo. Tato implementace může být upravena pro podporu jiných platforem. Je dostupná i implementace pro OpenStack.

Image Management

V distribuované infrastruktuře EGI Federated Cloud uživatelé potřebují způsob, jak efektivně organizovat a distribuovat obrazy pro virtuální stroje v rámci heterogenního prostředí. EGI Federated Cloud VM Image managent umožňuje uživatelům registrovat nové VM obrazy pro automatickou distribuci jednotlivým federovaným poskytovalům prostředků. Služba zastřešující tuto funkcionalitu je EGI Application Database (AppDB).

AppDB nabízí webové rozhrání uživatelům pro registraci a organizaci metadat VM obrazů a pro zařazení těchto obrazů jako součásti virtuálních organizací.

Preferovaný formát pro obrazy je otevřený standard DMTF Open Virtualization Format (OVF), který si uživatelé podle potřeby můžou převést na vhodný formát, který podporuje jejich platforma pro management cloudového prostředí.

3.3.3 Integrace do EGI Federated Cloud

S EGI Core infrastrukturou se pojí integrace do ní. Jednotliví poskytovalé musí mimo vlastních výpočetních kapacit nabídnout i integraci jistých služeb.

AAI

Federace identit je jedna ze základních částí kterékoliv infrastruktury s interoperabilitou. Bez federace identit uživatelé nemůžou přistupovat k infrastruktuře transparentně, jelikož by potřebovali specifické přihlašovací údaje ke každému poskytovateli v infrastruktuře. EGI kontroluje přístup k prostředkům skrze osobní X.509[9] certifikáty a pomocí konceptu tzv. Virtuálních Organizací (VO). Pro správu VO je používaný VOMS - Virtual Organization Membership Service.

Accounting

Užívání prostředků je zaznamenováno a integrováno do EGI Core Accounting systému. Existuje formát Cloud Usage Record (UR), který definuje data, která musí poskytovatel posílat centrálnímu registru EGI. Data UR jsou zašifrovaná veřejným klíčem EGI Accounting a podepsaná privátním klíčem hostujícího poskytovatele,

[9]V kryptografii standard pro systémy založený na veřejném klíči pro jednoduché podepisování

která jsou následně zaslána jako bezpečné data. EGI Accounting portál nabízí uživatelům, operátorům a administrátorům pohledy pro přístup k accounting datům ve strukturované či grafické podobě.

Information Discovery

EGI Information Discovery služba poskytuje uživatelům a nástrojům prostředky k zjištění dostupných prostředků v infrastruktuře. EGI provozuje informační systém založený na Berkeley Database Information Index (BDII) s hiearchickou strukturou distribuovanou přes celou infrastrukturu. Pro EGI Federated Cloud bylo vyvinuto schéma pro publikaci informací týkajících se dostupných poskytovatelů a jejich kapacit. Hlavním klientem EGI Information služby je AppDB, která zobrazuje poskytovatele prostředků nabízející specifické obrazy virtuálních strojů a detaily pro jejich využití.

Availability Monitoring

Každá služba v EGI infrastruktuře je monitorována pomocí Service Availability Monitoring (SAM). Poskytovatelé jednotlivých technologií musí poskytnout specifické sondy pro kontrolu funkcionality a dostupnosti služeb v souladu s Operational Level Agreements(OLA), které jsou zařízené v momentě, kdy se federovaný poskytovatel prostředků stane členem infrastruktury. Pro EGI Federated Cloud služby existuje několik monitorovacích sond jakou je klasický ping.

3.3.4 Vývoj EGI Federated Cloud

Infrastruktura EGI Federated Cloud se průběžně vyvíjí kvůli narůstající potřebě uspokojit požadavky na funkcionalitu formulované EGI uživateli a poskytovateli (např. Competence centra, Virtuální organizace, NGI apod.). V momentální fázi se tato evoluce dělí na tři části:

- Vývoj další funkcionality z platforem managementu cloudového prostředí skrze rozhrání standardu OCCI
- Vývoj mechanismu pro migraci virtuálních strojů mezi poskytovateli
- Poskytování základních message broker systémů, např. na bázi AMQP

Další požadovaná funkcionalita přes rozhrání OCCI je podpora vytváření tzv. snapshotů[10] běžící instance virtuálního stroje a podpora pro změnu přiřazeného prostředku k virtuálnímu stroji (např. změna velikosti operační paměti, zvýšení počtu dostupných jader CPU apod.). Tato funkcionalita je již dostupná ve všech platformách managementu cloudového prostředí v EGI Federated Cloud, ale chybí zde

[10]Kopie systémů v daný okamžik

podpora ve standardu OCCI a zejména zde chybí jednotný názor na to, jak by se tato funkcionalita měla jednotně zachovat.

Jedna z dalších požadovaných funkcionalit je podpora migrace virtuálních strojů mezi jednotlivými federovanými poskytovateli prostředků. Toto je poměrně náročná operace, obzvláště mezi různými cloudovými platformami.

EGI AppDB je aktuálně oficiálním katalogem obrazů a aplikací běžících na obrazech. Je centrálním bodem, kde uživatelé můžou vyhledávat potřebnou aplikaci nebo virtuální stroj, který má běžet v EGI Federated Cloud. Tím pádem je logické, že by AppDB se měla vyvinout tak, aby přešla z role katalogu na systém, který by sám uměl spouštět virtuální stroje a uměl tak nabídnout uživatelům správu virtuálních strojů přímo přes webové rozhrání místo toho, aby museli použít specializované nástroje.

3.3.5 Podpora uživatelů

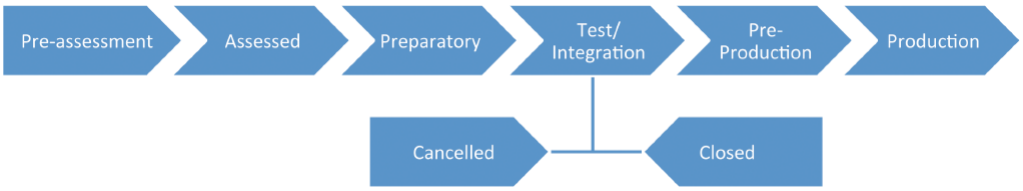
V EGI byla vyvinuta strategie podpory pro podporu komunit zajímající se o využití kapacit a prostředků EGI Federated Cloud. Poskytování srozumitelné dokumentace, nízká bariéra pro uživatele si vyzkoušet a zhodnotit služby a dobře definovaný proces podpory pro přiblížení use cases fázi produkce byly identifikovány jako klíčové faktory úspěšnosti garantující dlouhodobě udržitelnou infrastrukturu pro budoucí uživatele.

Od startu produkčních aktivit v květnu 2014 EGI vynaložilo nemalé úsilí pro tvorbu, sběr a vylepšení dokumentace týkající se cloudu, která je dostupná na otevřené wiki stránce Federated Cloud User Support. Dokumentace popisuje cloudové služby a obsahuje návody a FAQ, která vede uživatele k lepšímu pochopení nových paradigmat a definuje procedury krok po kroku k maximalizaci užitku při použití cloudu. EGI také dokumentuje aplikační modely úspěšných use cases k usnadnění adoptování budoucími uživateli. Spolu s dokumentací je ve vývoji i tréninkový program, který je zaměřen na použití infrastruktury budoucími uživateli.

Pro zjednodušení přístupu vědců k okamžitému vyzkoušení a zhodnocení EGI Federated Cloud byla vytvořena globální virtuální organizace (VO) podporovaná všemi poskytovateli prostředků ve federaci. VO se v tomhle případě se chová jako inkubátor pro nové use cases, kde uživatelé můžou prototypovat a validovat jejich aplikace po dobu až 6 měsíců. Tato VO velkou mírou snížila bariéru pro vstup uživatelů do cloudu odstraněním potřeby pro dlouhé administrativní procesy související s registrací do existující VO, obzvláště pro uživatele bez předchozí zkušenosti s gridem a absencí podpory z jeho komunity.

To je obzvláště užitečné pro uživatele v tom smyslu, že si můžou ověřit, zda-li jejich use case má využití pro cloud a pokud ano, tak můžou okamžitě pokračovat na

integraci jejich nástrojů, služeb a aplikací bez administrativní režie. Vedle toho EGI nabízí i sadu obrazů virtuálních strojů, které jsou okamžitě připraveny k použití v AppDB pro zrychlení nasazení bez potřeby tvorby vlastních obrazů. Tyto virtuální stroje jsou dostupné u všech poskytovatelů ve federaci a jsou konfigurované tak, aby bylo na nich možné využívat všechny kapacity EGI Federated Cloud. Obrazy VM pokrývají všechny běžně dostupné operační systémy, kde si uživatelé mohou nainstalovat vlastní aplikace.



Obr. 3.6: Schéma workflow podpory uživatelů [8]

3.4 NGI

National Grid Infrastructures jsou národní gridové infrastruktury jednotlivých států, které jsou zapojené do infrastruktury EGI. Za Českou Republiku reprezentuje NGI MetaCentrum, provozované sdružením CESNET.

3.4.1 MetaCentrum

[MetaCentrum] Počátky gridové infrastruktury MetaCentrum se datují k roku 1996, tedy až k samotnému vzniku sdružení CESNET, které MetaCentrum nyní provozuje. Zahrnuje řadu výpočetních clusterů umístěných v několika lokalitách a patřících různým subjektům (ve vlastnictví CESNET se nachází asi polovina strojů MetaCentra). Všechny jsou integrovány do jednotného prostředí se společnou správou uživatelů a úloh.

MetaCentrum vystupuje v roli České Národní Gridové Infrastruktury (NGI-CZ), oficiálně uznávané národní součástí Evropské Gridové Infrastruktury (EGI). Hlavním úkolem je další rozšiřování a koordinace plnohodnotné národní gridové infrastruktury v České republice vhodným propojením stávajících a nově pořizovaných výpočetních a úložných prostředků akademické komunity v ČR. Vytvoření virtuálního pracovního prostředí MetaCentrum přispívá k podstatně efektivnějšímu využití instalované techniky, umožňuje využití dostupných výpočetních zdrojů pro řešení velmi náročných výpočetních úloh, jejichž zvládnutí je nad možností samostatného pracoviště v ČR. Zájemcům z akademických a výzkumných institucí nabízí MetaCentrum prostředí pro (spolu)práci v oblasti výpočtů a ukládání dat nejen na území ČR, ale i v mezinárodním měřítku (podpora integrace do mezinárodního výzkumného prostoru (ERA) a napojení do Evropské Gridové Infrastruktury EGI).

MetaCentrum koordinuje pořizování nákladného programového a aplikačního vybavení, zajišťuje centrální nákup a správu licencí vybraného licencovaného SW. Zároveň se MetaCentrum aktivně podílí na výzkumu a vývoji nezbytného k zajištění optimální funkcionality, bezpečnosti a vysokého výkonu celé infrastruktury. Naši odborníci jsou zapojeni do odpovídajících mezinárodních projektů. Zájemci mohou požádat o členství v některé z námi spravovaných virtuálních organizací nebo mohou požádat o vytvoření vlastní virtuální organizace. [9]

MetaCentrum nabízí široké softwarové portfolio aplikačních programů, zaměřující se na několik tematických okruhů:

- Výpočetní chemie/Molekulové modelování - Amber, Babel, Gaussian GaussView
- Technické a materiálové simulace - ANSYS, Fluent, MSC.Marc, WIEN2k
- Matematické a statistické modelování - Maple, Matlab, SNNS, gridMathematica, R, Vista
- Vývojářské nástroje a prostředí - SGI, PGI CDK, TotalView, Vampir, Paradise, SICStus Prolog

Obr. 3.7: Mapa rozmístění hardware v rámci MetaCentra [9]

4 ANALÝZA SOUČASNÉHO STAVU

V analýze současného stavu je třeba se zaměřit primárně na samotnou distribuovanou službu, se kterou budeme pracovat. Zejména je třeba objasnit si, zda-li službu je vůbec možné přenést do jiného cloudového prostředí, než ve kterém právě pracuje. To zahrnuje analýzu jak aktuálního cloudového prostředí META-pipe, tak analýzu technologií využívaných právě v této službě.

4.1 Implementace a integrace META-pipe v NeLS

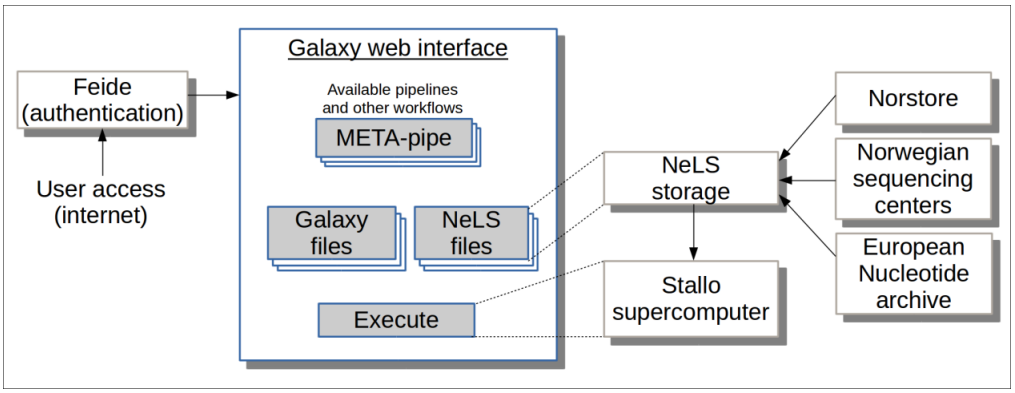
META-pipe je navržená tak, aby byla schopna škálovat na obrovských metagenomických datových souborech. Momentálně je integrovaná na platformách METAREP a Galaxy, aby mohla nabídnout flexibilní, ale uživatelům známé prostředí. V NeLS[1] je META-pipe nasazená na národní úrovni pro zjednodušení přístupu k META-pipe a využití národního úložiště (NorStore) a high performance výpočetních zdrojů (Notur).

4.1.1 Používání META-pipe

META-pipe je jedna z několika pipeline využívaných v NeLS. NeLS služby používají standardizované uživatelské prostředí pro přihlášení, management dat a workflow management. K tomu je využívané prostředí Galaxy. Pro autentizaci a pro autorizaci je využitý poskytovatel Feide - federovaný identity provider pro norské univerzity. To umožňuje se přihlásit norským studentům se přihlásit účtem své školy. Uživatelé z jiných zemí se můžou přihlásit separátně pomocí Nels IdP. Po přihlášení je umožněno uživateli nahrát vstupní data několika způsoby:

- Nahrání souboru přes webový prohlížeč
- Nahrání přes poskytnutou URL
- Pro větší datové soubory je umožněno propojí s úložištěm NorStore

[1] Norwegian e-Infrastructure for Life Sciences - projekt vytvořený ELIXIR Norway pro podporu life sciences



Obr. 4.1: Integrace META-pipe v NeLS [1]

4.1.2 Galaxy

Galaxy je primárně webová platforma pro integraci vědeckých postupů, dat a jejich publikaci zaměřená především pro bioinformatiku. Jejím primárním cílem jsou výzkumní vědci bez znalosti nebo zkušenosti s programováním nebo systémové administrace.

Galaxy vznikla jako projekt na Penn State University. Byla vyvíjena především pro genomický výzkum, ale je možné ji využít pro libovolné bioinformatické odvětví jako je proteomika, metagenomika nebo epigenomika.

Galaxy je volně dostupná jako svobodný software na veřejném repozitáři GitHub. Galaxy má nejenom privátní, ale i veřejné servery s velkou výpočetní kapacitou, která je dostupná bioinformatikům z celého světa. Momentálně existuje asi 80 veřejných serverů se statisíci jádry. Výhodou je to, že každý, kdo má nějakou výpočetní kapacitu se může přidat a poskytnou své zdroje vědecké komunitě.

META-pipe workflow v Galaxy se skládá z preprocessing nástrojů, nástrojů taxonomické klasifikace a nástrojem pro funkcionální analýzu. Výstupem těchto nástrojů je soubor ve formátu zip, která obsahuje všechny soubory funkcionální anotace ve formátu, který byl zvolen. Taxonomická klasifikace může být zobrazena v programu Krona v prohlížeči nebo může být stažena pro otevření v samostatných programech.

Funkcionální analýza byla implementovaná jako nástroj v Galaxy především kvůli tomu, že byl již dostupný workflow manager pro paralelní spouštění clusterů. Tento nástroj má tu nevýhodu, že je tzv. black box - uživatel vidí pouze jeho vstupy a výstupy, ale ne jeho přímou implementaci. Nicméně alternativa k tomu je redesign spouštění pipeline funkcionální analýzy na clusterech.

Galaxy běží na virtuálním stroji se specifikacemi 32 GB RAM, 2 CPU jádra a 4 TB úložištěm[2], což je poměrně nevýkonný stroj, jelikož se zpracování většiny dat odkládá na superpočítač Stallo.

[2]Toto úložiště je využívané pouze k ukládání dat uživatelů

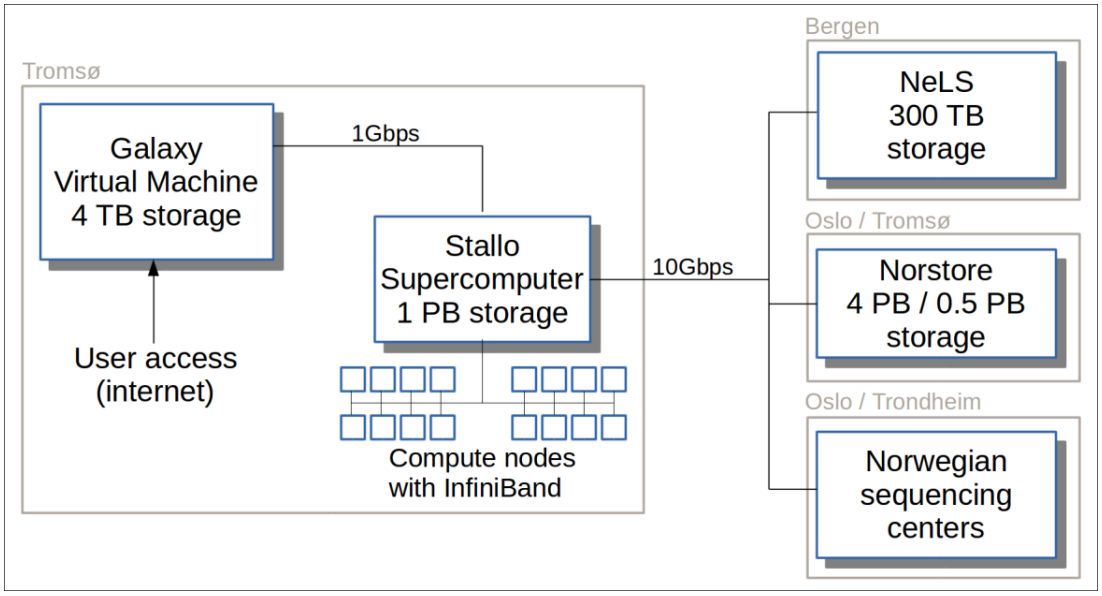
4.1.3 Integrace úložiště

S Galaxy je integrované se systémy úložiště superpočítače Stallo a NeLS úložištěm v Bergenu. Všechny úložiště jsou aktivně využívané. Pro dlouhodobé ukládání by uživatelé měli používat úložiště NorStore, které je k tomu uzpůsobeno.

Postup pro uživatele, který si chce archivovat jejich vstup a výsledek analýzy je následující:

1. Vložení surových dat do NorStore archívu. Data mohou být zkopírovány přímo z dočasného úložiště
2. Kopírování surových dat do projektového prostoru uživatele v NeLS úložném systému
3. Kopírování dat z NeLS do Galaxy pomocí Galaxy nástrojů vyvinuté v NeLS projektu
4. Kopírování dat z Galaxy do superpočítače, kde jsou data čteny uzly v superpočítači
5. Spuštění META-pipe workflow na superpočítači
6. Zkopírování výsledků do Galaxy
7. Zkopírování výsledků z Galaxy do NeLS úložiště
8. Archivace výsledků do NorStore

Data v Galaxy se berou jako hlavní kopie a kopie v ostatních úložištích se berou jako záložní repliky. Je tedy na uživateli, aby si zajistil, že jsou data archivovány pro dlouhodobé skladování.

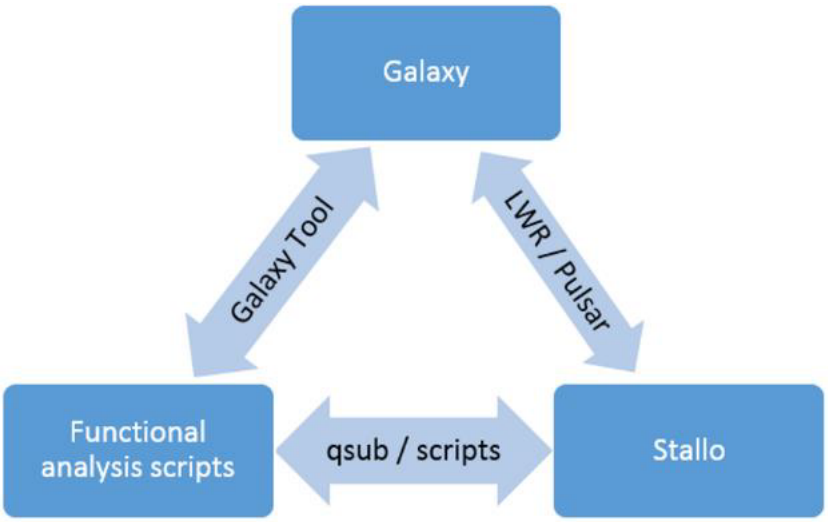


Obr. 4.2: Integrace úložiště v Galaxy [1]

4.1.4 Integrace superpočítače

META-pipe je integrovaná přes Galaxy pomocí TORQUE[3]. Galaxy spouští META-pipe na jednom či několika z více jak 800 výpočetních clusterů dostupných na superpočítači Stallo.

Stallo v současnosti obsluhuje 18.144 CPU jader, 26.2 TB RAM a 2 PB úložného prostoru. Uzly jsou propojené pomoc InfiniBandu s rychlostí 40 Gb/s a běží na softwaru skládajícího se z CentOS 6.2 a PBS/TORQUE plánovače. Stallo umožňuje zpracování největších metagenomických datových souborů a jejich paralelní běh. Pro menší datové soubory je dostupný stroj s 12 CPU jádry, 256 GB RAM a 5.5 TB úložného prostoru.

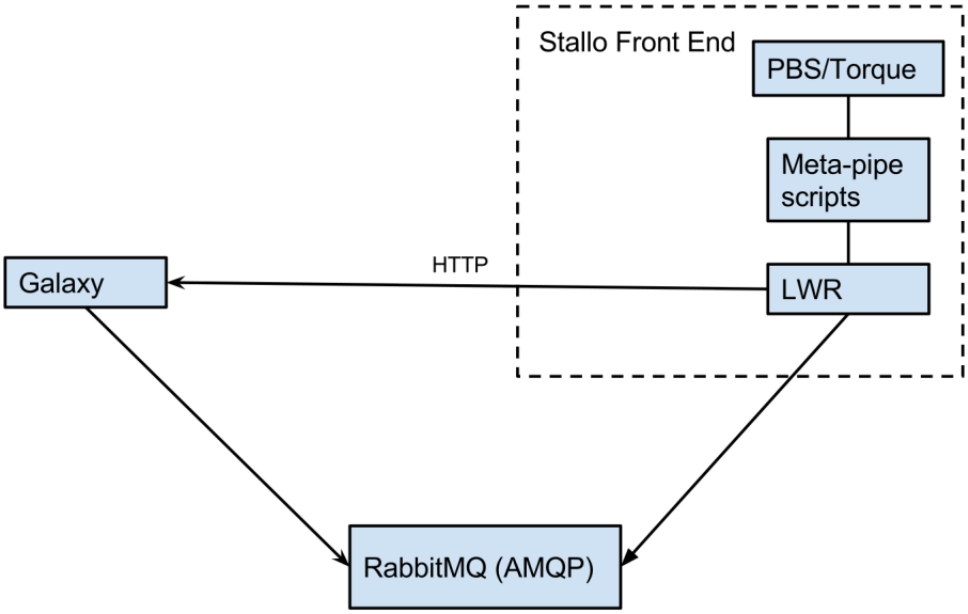


Obr. 4.3: Integrace superpočítače a Galaxy [1]

Pro nástroje funkcionální analýzy v META-pipe byl implementovaný workflow manager zvaný LWR, který řeší data management a zpracování paralelních úloh. Tento manager byl psán ručně v Perlu, nicméně postupem času rostl a nyní zahrnuje víc jak 10 tisíc řádků zdrojového kódu. Toto řešení bylo zvoleno proto, že META-pipe jen velmi specifický služba a kód musel být napsaný přesně pro ni.

LWR slouží ke spouštění META-pipe úloh paralelně na Stallo superpočítači. LWR komunikuje přes Galaxy API a RabbitMQ[4].

[3]Terascale Open-source Resource and QUEue Manager je plánovač pro distrubované systémy
[4]Open-source message broker fungující na AMQP protokolu určený pro nasazení v HPC



Obr. 4.4: Integrace LWR v superpočítači Stallo [1]

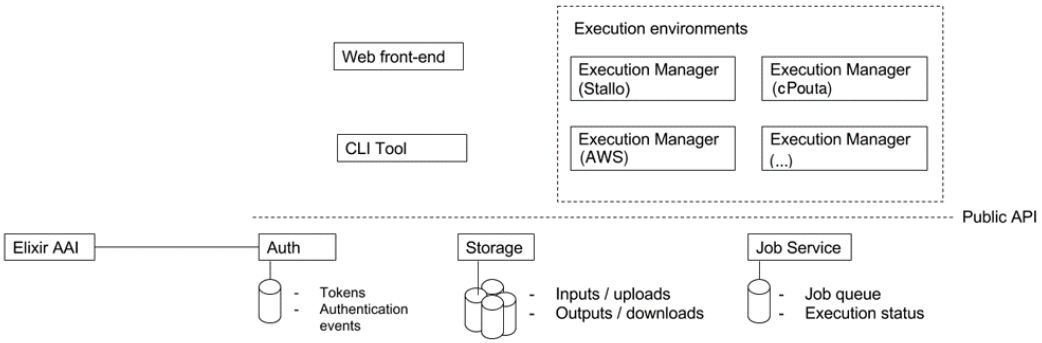
4.2 Obecná integrace META-pipe a běh klienta

V předchozí kapitole jsme se zabývali implementací a integrací META-pipe v prostředí NeLS. Toto prostředí má své specifika díky integraci národní infrastruktury a superpočítače Stallo, nicméně META-pipe je navržená tak, aby běžela i v jiném distribuovaném prostředí. Důležitým cílem designu backendu META-pipe je přenositelnost executing managerů. To je umožněno tím, že stavy běžících úloh jsou uložené na centrálních serverech.

4.2.1 Server

Na jednom ze serverů META-pipe běží webová služba, který buď přes UI nebo REST API přijímá úlohy pro zpracování metagenomických dat. To může být jak Galaxy, tak čerstvě vyvinutá služba implementující API pro zpracování úloh zvaná META-pipe web application. Tato služba má tu výhodu, že je napsaná přímo pro META-pipe a není tak omezená platformou Galaxy, nicméně je velmi jednoduchá a nenabízí tolik nástrojů.

Úlohy se rozlišují podle tzv. job tags, které určují, kterým clusterům mají úlohy připadnout. Úlohy může spouštět uživatel, který je přihlášený k Galaxy přes NeLS IdP, nebo skrze ELIXIR AAI v META-pipe web application. Přihlašování skrze ELIXIR AAI má tu výhodu, že se lze přihlásit účtem libovolné instituce zapojené do ELIXIRu, což je velké množství univerzit a výzkumných institucí.



Obr. 4.5: Architektura serveru META-pipe [10]

4.2.2 Klient

Clustery se skládají s jednoho tzv. master uzlu, který ovládá několik dalších tzv. worker uzlů[5]. Master uzel přímo komunikuje s centrálními META-pipe servery, od kterých přebírá úlohy. Na master uzlu běží instance software Apache Spark, který úlohy spouští a přiděluje je jednotlivým worker uzlům. Klient je spuštěný s určitou "job tag". To zajišťuje, že se klient zpracovává pouze ty úlohy, které jsou mu určené.

Clustery jsou samostatná jednotka, která může být a nemusí být přímo připojena k internetu. Pro oddělené vnitřní sítě od internetu je možné použít tzv. bastion hostu, což je počítač specificky navržený na odolání útoku zvenčí. Na něm běží proxy, skrz kterou komunikuje master uzel clusteru s vnější sítí.

Klientské prostředí teda můžeme rozdělit na tyto uzly:

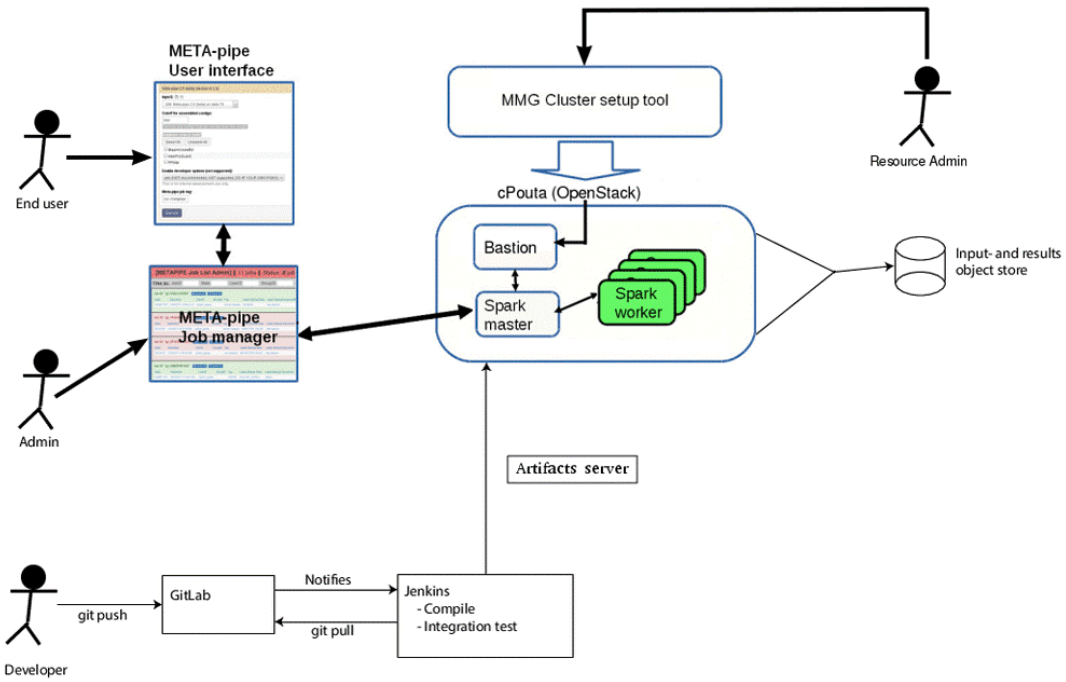
- Bastion
- Cluster Master
- Cluster Worker

META-pipe vyžaduje sdílené úložiště pro worker uzly, v tomhle případě NFS[6] pro čtení a zápis dočasných výpočtených dat včetně předběžných výsledků. Toto úložiště může být jak interní úložiště master uzlu, tak virtuální úložiště připojené k master uzlu. Druhé řešení je jednodušší pro provoz, jelikož se takové úložiště může nachystat před během samotného klienta a zrychluje to proces samotného výpočtu.

Jelikož nasazení výpočetního clusteru ručným spouštěním virtuálních strojů a instalací softwaru na každý z nich je časově velmi náročná práce, byl vyvinut nástroj MMG Cluster setup, který toto nasazení automatizuje.

[5]Master/slave architektura

[6]Network File System je internetový protokol pro vzdálený přístup k souborům přes síť



Obr. 4.6: Architektura klienta META-pipe [10]

4.2.3 MMG Cluster setup

Tento nástroj byl taktéž vyvinut Univerzitou v Tromsu pro zjednodušení nasazení META-pipe klientů. Byl napsán v Javě a funguje jako konzolová aplikace, která je schopná kompletní přípravy prostředí pro běh klienta META-pipe. Tento nástroj je určený především pro infrastrukturu CSC cPouta, nicméně lze použít pro libovolnou OpenStack[7] platformu.

MMG Cluster setup nástroj provádí tyto kroky:

- Příprava OpenStack prostředí
- Příprava software META-pipe pro běh v clusteru
- Příprava samotného clusteru, virtuálních disků, zabezpeční a NFS úložiště pro využití v clusteru
- Konfigurace Spark serveru a test funkčnosti Sparku v clusteru
- Příprava, validace a běh processing softwaru v clusteru
- Vyčištění a úklid existujícího clusteru

Pro přípravu OpenStack prostředí a instalaci softwaru na jednotlivé uzly je využit open-source nástroje Ansible.

[7]Open-source softwarová platforma vyvíjená v programovacím jazyce Python, využívaná především jako IaaS. Vznikla v roce 2010 jako společný projekt firmy Rackspace Hosting a NASA

5 NÁVRH ŘEŠENÍ

V předchozích kapitolách jsme si důkladně analyzovali META-pipe, její komponenty a funkce.

Úkolem této práce je zajistit přenositelnost META-pipe mezi různými distribuovanými prostředí. Vybrané prostředí, do kterého se bude přenášet META-pipe se nazývá EGI Federated Cloud. Zde se používá rozhrání OCCI, které musí být implementované, abychom mohli využívat nabízené prostředky.

Jelikož je META-pipe navržená tak, aby byla spustitelná s minimálními úpravami v libovolném distribuovaném prostředí, nebude třeba přepisovat větší části zdrojového kódu. Pro tvorbu výpočetního clusteru je standardně používán nástroj Ansible, nicméně ten nelze využít jako jediný. Jako alternativa, či spíš doplněk byl zvolen program Terraform.

Je třeba napsat i program, který bude tyto nástroje využívat, jelikož tyto nástroje jsou automatizované, ale je třeba je správně nakonfigurovat a spouštět. K tomu bude sloužit aplikace napsaná v programovacím jazyce Python. Ten byl zvolen z několika důvodů, mezi ty důležité patří to, že je dostupný na všech operačních systémech a je široce používaný ve vědecké komunitě.

Všechny programy a nástroje používáné v této práci jsou open-source, což znamená, že mají volně dostupný zdrojový kód včetně samotné META-pipe.

5.1 Open Cloud Computing Interface

Open Cloud Computing Interface (OCCI) je otevřený standard pro cloudové infrastruktury ve formě REST API. Tento standard vznikl pro usnadnění interoperability různých cloudových infrastruktur - OCCI nabízí jednotné rozhrání, skrz které lze přistupovat ke zdrojům v kompatibilním cloudu.

OCCI bylo původně vytvořeno jako API pro vzdálený management IaaS služeb, nicméně současný standard umožňuje použití i pro PaaS[1] a SaaS[2] služby. Aby OCCI bylo modulární a rozšířitelné, OCCI specifikace jsou uvolněné jako sada komplementárních dokumentů, které dohromady tvoří kompletní specifikaci. Tyto dokumenty jsou dělené do čtyř kategorií:

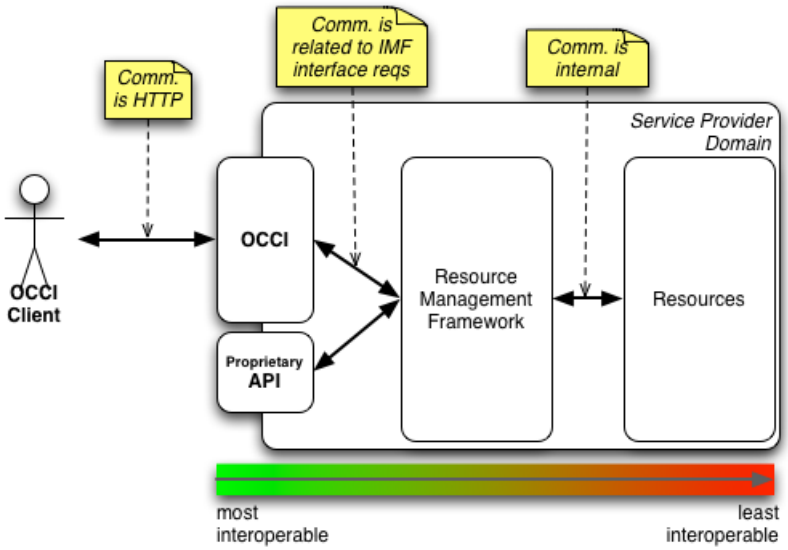
- OCCI Core specifikace se skládají z jednoho dokumentu definující OCCI Core model. OCCI interakce se tvoří přes tzv. renderingy (včetně sdružených aktivit) a jsou rozšířitelné přes tzv. extensions
- OCCI Protocol specifikace se skládají z několika dokumentů, kde každý popisuje způsob, jak model může komunikovat přes specifikovaný protokol (např.

[1] Platform as a service

[2] Software as a service

HTTP, AMQP apod.). Vícero protokolů může komunikovat s jednou instací OCCI Core modelu

- OCCI Rendering specifikace se taktéž skládají z vícero dokumentů, kde každý popisuje konkrétní rendering OCCI Core modelu. Vícero renderingů může komunikovat se stejnou instací OCCI Core modelu a automaticky podporují libovolné dodatky k modelu, které dodržují pravidla definovaná v OCCI Core
- OCCI Extension specifikace definuje jednotlivé rozšíření v OCCI Core modelu



Obr. 5.1: Umístění OCCI v rámci cloudové infrastruktury [11]

5.1.1 rOCCI-cli

rOCCI-cli je konzolový klient pro OCCI napsaný v programovacím jazyce Ruby. Umožňuje přímou interakci s OCCI endpointy v EGI Federated Cloud. Jelikož podporuje všechny funkce OCCI, bude jej možné použít do Terraform OCCI pluginu.

rOCCI-cli se může instalovat jak z AppDB, tak jako Ruby gem[3]. Výhodou instalace z AppDB je integrace voms-clients[4]. Tato instalace funguje jako virtuální stroj, který lze nainstalovat libovolným virtualizačním nástrojem, např. VirtualBox.

[3]Ruby balíčky jsou zvané gems. RubyGems je package manager, který s nimi pracuje
[4]Klient pro správu autentizace vůči virtuálním organizacím

5.2 Terraform

Terraform je open-source nástroj pro tzv. "Infrastructure as a code"[5] napsaný v programovacím jazyce Go[6] a vyvíjený firmou Hashicorp.

Terraform umožňuje definovat a tvořit virtuální datacentra pomocí souborů ve formátu HCL[7] nebo JSON. Podporuje velkou část veřejných cloudových infrastruktur jako Amazon Web Services, Google Cloud Platform nebo Microsoft Azure i privátní cloudy jako OpenStack.

Terraform je primárně určený jako nástroj pro budování, změnu a verzování infrastruktury bezpečně a efektivně. Konfigurační soubory ve formátu HCL popisují Terraformu komponenty nutné k běhu jedné aplikaci či celého datacentra. Terraform generuje spouštěcí plán popisující co má v plánu dělat, aby dosáhl vytyčeného cíle a následně plán spustí pro stavbu infrastruktury. Jak se v průběhu mění konfigurace, Terraform je schopný určit co se změnilo a podle toho se zachovat. Infrastruktury, které Terraform umí managovat jsou např. nízkoúrovňové komponenty jako instance virtuálních strojů, úložiště, sítě stejně jako vysokoúrovňové komponenty jako DNS záznamy, SaaS služby apod.

Klíčové prvky Terraformu jsou:

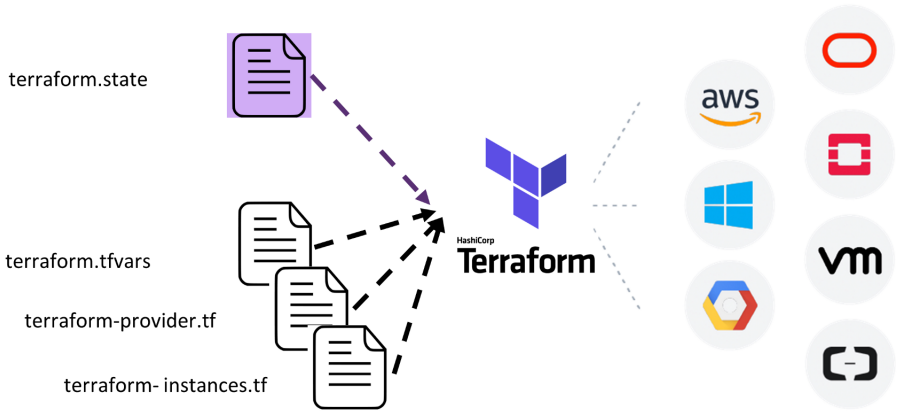
- Infrastructure as a Code - Infrastruktura je definovaná pomocí vysokoúrovňové syntaxe. To umožňuje verzovat si infrastrukturu datacentra a chovat se k ní jako ke kterémukoliv jinému zdrojovému kódu. To umožňuje například infrastrukturu sdílet a znovupoužít.
- Execution plans - Terraform má plánovací krok, kde generuje tzv. execution plan. Ten určuje, co má v plánu Terraform udělat při běhu, čímž se eliminují nepříjemná překvapení před úpravou infrastruktury
- Resource graph - Terraform tvoří graf všech prostředků a paralelizuje tvorbu a modifikaci všech nezávislých prostředků.
- Change Automation - Komplexní změny mohou být provedeny na infrastruktuře s minimální účastí uživatele.

Jelikož podpora pro jednotlivé cloudové infrastruktury se skládá v Terraformu z modulů, je třeba napsat modul i pro OCCI. Moduly se dají psát v jazyce Golang, přičemž je lze psát jako jednoduché volání REST API či v něm obalit jednoduchý spustitelný klient v příkazové řádce.

[5] Infrastructrue as a code je paradigma udržování infrastruktury datacentra v souborech, které jsou strojově čitelné

[6] Go, též zvaný Golang je programovací jazyk vyvinut společností Google

[7] Hashicorp Configuration Language



Obr. 5.2: Architektura Terraformu [12]

5.2.1 Porovnání s jinými nástroji

Terraform byl porovnán s jinými podobnými nástroji ke správě infrastruktury jako infrastructure as a code:

	Terraform	Chef	Puppet	Ansible	SaltStack	CloudFormation
Kód	Open source	Open source	Open source	Open source	Open source	Closed source
Cloud?	Vše	Vše	Vše	Vše	Vše	Pouze AWS
Typ	Orchestrace	Konfigurace	Konfigurace	Konfigurace	Konfigurace	Orchestrace
Infrastruktura	Immutable[8]	Mutable	Mutable	Mutable	Mutable	Immutable
Jazyk	Deklarativní	Procedurální	Deklarativní	Procedurální	Deklarativní	Deklarativní
Architektura	Klient	Klient/Server	Klient/Server	Klient	Klient/Server	Klient

Tab. 5.1: Porovnání Terraformu s jinými infrastructure as a code nástroji, Zdroj: autor

Terraform byl pro tuto práci především jeho podpoře více cloudových infrastruktur,modularitě, schopností orchestrace a klientskou architekturou. V původním nasazením v Norsku se používal Ansible na orchestraci i konfiguraci, v téhle práci se bude používat Terraform pro orchestraci a Ansible ke konfiguraci vzhledem k nevhodnosti Ansible pro orchestraci cloudových infrastruktur na bázi OCCI.

[8]Immutabilita je vlastnost objektu, kdy po jeho vytvoření již nelze měnit jeho vnitřní hodnoty

5.3 Ansible

Ansible je open-source nástroj využívaný především jako konfigurační nástroj pro instalaci aplikací na fyzické i virtuální stroje. Běží na většině Unix-like systémů a umí konfigurovat jak Unix-like, tak Windows stroje. Jeho hlavní síla spočívá v tom, že je tzv. agentless - nepotřebuje k fungování žádný program běžící na cílovém stroji kromě SSH[9] serveru.

Nástroj byl v roce 2015 odkoupen firmout Red Hat. Je součástí distribucí založených na Red Hat Enterprise Linuxu jako je např. Fedora, CentOS, Scientific Linux apod.

Jako konfigurační nástroj je v DevOps[10] často využíván v kombinaci s orchestračními nástroji, především Terraformem, což je kombinace, která se bude právě využívat v této práci.

5.3.1 Architektura

Oproti většině nástrojů pro management konfigurace (např. Puppet) Ansible nevyžaduje jediný běžící stroj pro začátek orchestrace. Ansible pracuje na několika systémech v infrastruktře selekcí částí tzv. Inventory, která je uložená jako verzovatelný ASCII textový soubor. Tímto se dá říct, že Ansible taktéž jako Terraform funguje na paradigmatu Infrastructure as a code. Tento inventář je nejenom konfigurovatelný, ale zároveň je možné využívat vícero inventářů ve stejný čas a tento inventář získavat např. z cloudu nebo v jiném formátu.

Každý stroj s nainstalovaným Ansible má stejnou funkcionalitu jako jiný stroj, což zjednodušuje rekonstrukci infrastuktury vzhledem k absenci centrálního serveru. Kontrolované uzly jsou managovány jedním z kontrolních strojů, typicky přes SSH. Pokud je potřeba využít citlivých dat, je možné použít Ansible Vault - šifrované úložiště souborů.

Designové cíle

Mezi designové cíle Ansible patří:

- Minimalismus - Managované systémy by neměly vyžadovat další požadavky na prostředí
- Konzistence - Ansible tvoří konzistencí infrastrukturu
- Bezpečnost - Ansible nevyžaduje žádné další běžící programy, čímž snižuje množství vektorů útoků

[9] Secure shell - kryptografický síťový protokol pro ovládání přes nezabezpečenou síť

[10] Přístup k vývoji software a infrastruktury, který zdůrazňuje spolupráci mezi vývojářem a systémovými administrátory

- Vysoká spolehlivost - Při správné konfiguraci je Ansible idempotentní[11]

Moduly

Moduly jsou samostatné a lze je psát ve standardním skriptovacím jazyce jako je Pytho, Ruby, Perl atd. Jeden z požadavků na moduly je idempotence.

Konfigurace inventáře

Inventář je popisem uzlů, ke kterým má Ansible přístup. Ve výchozím nastavení je inventář popsaný konfiguračním souborem v INI nebo YAML formátu v lokaci `/etc/ansible/hosts`. Konfigurační soubor obsahuje seznam buď IP adres nebo hostnames všech uzlů. Tyto uzly mohou být zařazené do skupin.

Playbooky

Playbooky jsou soubory ve formátu YAML[12], která vyjadřují konfiguraci, nasazení a orchestraci v Ansible a dovolují provádět operace na managovaných uzlech. Každý playbook je namapován na skupinu hostů pomocí sady rolí. Každá role je reprezentována voláním Ansible úkonů.

Ansible Tower

Ansible Tower je REST API a webová služba, která umožňuje využívání Ansible IT týmům s rozdílnou úrovní dovedností. Automatizuje úkony Ansible, které by se jinak musely spouštět ručně.

Tower je komerční produkt podporovaný firmou Red Hat derivovaný z projektu AWX, který je open-source od září 2017.

[11] Idempotence je vlastnost operací, kterým se při opakovaném spouštění nemění stav

[12] YAML Ain't Markup Language je jazyk pro serializaci dat srozumitelný lidem, typicky používaný pro konfigurační soubory

5.4 Python

Python je interpretovaný[13] programovací jazyk. Designová filosofie Pythonu se zaměřuje na čitelnost kódu a jednoduchosti programování. Jeho jazykové konstrukce a objektově orientovaný přístup cílí na zjednodušení psaní čistého, logického kódu jako pro malé, tak velké projekty.

Python je dynamický typovaný[14] a využívá garbage collectoru[15]. Podporuje několik programovacích paradigmat jako procedurální, objektově orientované či funkcionální programování. Jeho největší síla spočívá ve standardní knihovně, která je součástí základní instalace programu a která stačí na většinu využití bez přídavných knihoven. To velice usnadňuje nasazení programů psaných Pythonu uživatelům.

V rámci práce se bude psát nástroj jako sada skriptů, kde každý bude tvořit jednu z funkcionalit:

- Založení clusteru
- Běh clusteru
- Zastavení clusteru
- Uklizení clusteru

Tyto skripty půjdou použít v sérii - hlavní skript například bude moct založit a spustit cluster zároveň.

[13]Interpretovaný jazyk je typ programovacího jazyka, ve kterém se jeho kód spouští přímo bez předchozí kompilace do strojového jazyka

[14]Typová kontrola proměnných v takovém jazyce probíhá až při běhu místo při kompilaci

[15]Typ automatického řízení práce s pamětí. Programátor nemusí přímo uvolňovat paměť, ale dělá to za něj samotný jazyk

5.5 MMG Cluster Setup - CESNET

V předchozích podkapitolách jsme si sjednotili technologie, které budeme používat při psaní nástroje zvaného MMG Cluster Setup - CESNET. Název nástroje vychází z původního MMG Cluster Setup, jelikož z něj přebírá velkou část Ansible playbooků, které instalují potřebný software. Přebírá z něj částečně i architekturu a funkcionalitu. Ne všechna funkcionalita byla implementovaná i zde (např. tvorba Bastion hostu) z důvodu nepraktičnosti nebo nepotřebnosti.

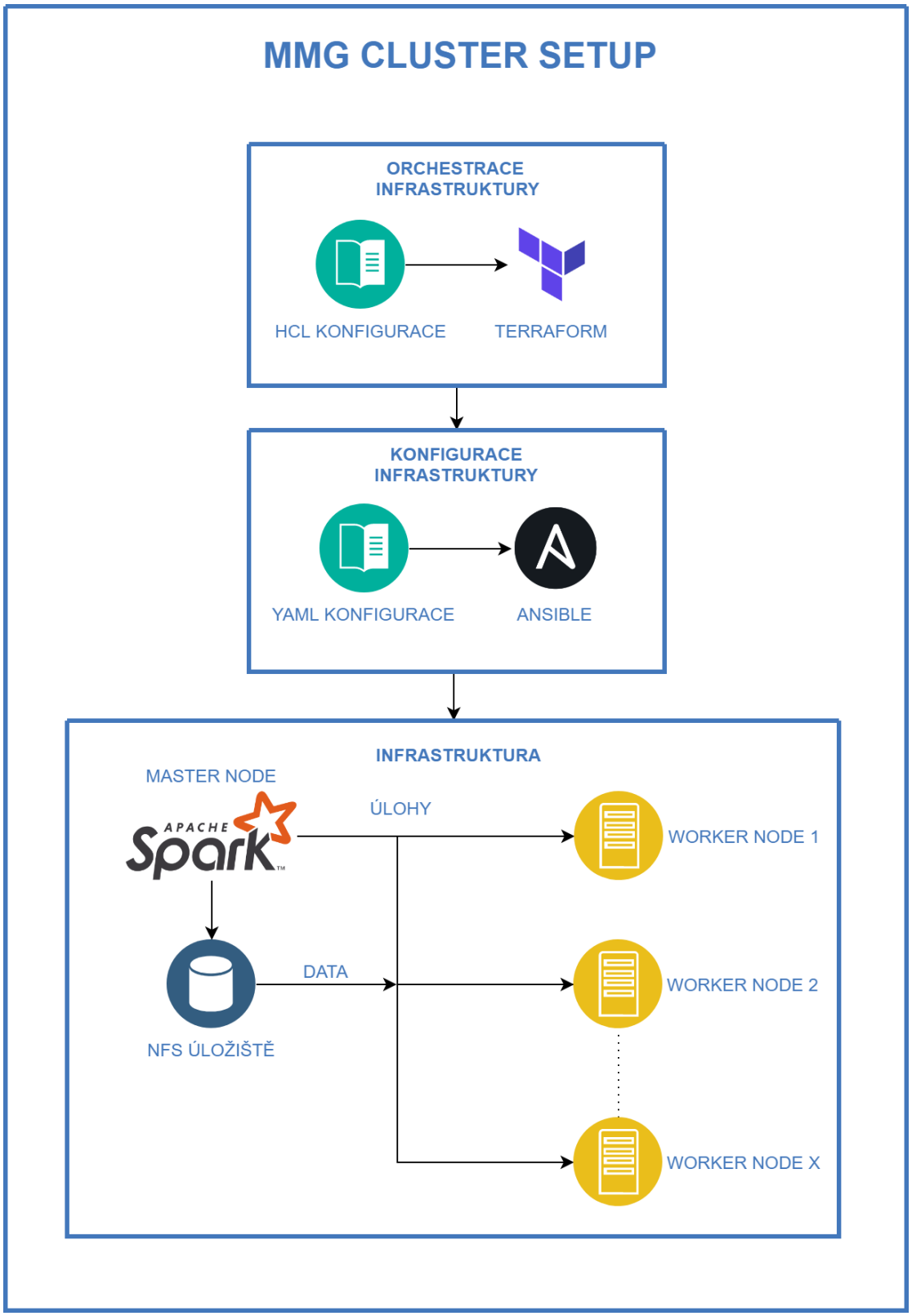
Jelikož zadáním práce je přenositelnost v rámci EGI Federated Cloud, je třeba použít orchestrační nástroj s podporou OCCI. Tím bude Terraform, pro který se napíše modul podporující OCCI. Ten bude obalovat klienta rOCCI pro zrychlení implementace a nasazení.

Po orchestraci clusteru přichází na řadu jeho konfigurace. Pro ten je využit Ansible a jeho playbooky. Jelikož se neinstaluje několik homogenních strojů, ale jeden master uzel a jeden až teoreticky nekonečno worker uzlů, je třeba si definovat, co se bude na který stroj instalovat. Pro to existují tzv. roles v playboocích, které podle nich určí, jaké playbooky se mají na jednotlivé stroje aplikovat.

Po těchto krocích je příprava clusteru hotová a můžou se začít pouštět úlohy na něm. Všechno je obalené v Python skriptech, které jsou spustitelné na libovolném uživatelském stroji s nainstalovaným Pythonem. Většina distribucí Linuxu a macOS již mají Python nainstalovaný v základu, v OS Windows je třeba jej nainstalovat zvlášť.

Kroky, které je třeba udělat pro spuštění clusteru jsou tedy následující:

1. Konfigurace Terraformu
2. Konfigurace Ansible
3. Spuštění Terraformu
4. Spuštění Ansible
5. Spuštění Spark hosta na master uzlu
6. Poslání úlohy přes centrální META-pipe server



Obr. 5.3: MMG Cluster Setup - CESNET tool, Zdroj: autor

6 IMPLEMENTACE ŘEŠENÍ

V této kapitole se bude práce zabývat konkrétní implementací řešení běhu METApipe v EGI Federated Cloud.

V návrhové kapitole byly určené technologie a postupy, kterým se bude implementace řídit. Kroky vedoucí k funkční implementaci budou provedeny v tomhle pořadí:

1. Vývoj Terraform OCCI modulu
2. Úprava Ansible playbooků
3. Vývoj MMG Cluster Setup nástroje

Každý z kroků vyžaduje použití jiného přístupu k vývoji vzhledem k značně rozdílné technologii. Terraform je psaný v Go, který je kompilovaným jazykem a nedynamickým typováním. Konfigurace Ansible playbooků je plně v YAML souborech s INI formátem inventáře. MMG Cluster setup je napsaný v Pythonu, který je interpretovaný, tudíž nekompilovaný a dynamicky typovaný.

6.1 Terraform OCCI modul

Terraform je psaný v programovacím jazyce Go, též zvaný Golang. Stejný jazyk se využívá i pro jeho moduly. Moduly se kompilují mimo hlavní binární soubor Terraformu a příkladají se při běhu aplikace dodatečně.

Terraform moduly mají jistou strukturu, kterou je potřeba dodržet. Musí mít několik vlastností, mimo jiné:

- Název
- Resource, který se modulem tvoří jelikož moduly budou tvořit OCCI virtuální stroje, resource se bude nazývat `occi_virtual_machine`
- Funkce, které resource nabízí. Terraform moduly vychází z CRUD[1] modelu. To tedy obnáší funkce:
  - Create
  - Read
  - Update
  - Delete

Tyto funkce jsou povinné a tvoří základní operace orchestrace. Pro jednotlivé operace se bude používat příslušný příkaz z příkazového klienta rOCCI.

[1] Create, Read, Update, Delete operace nad záznamem v trvalém úložišti

6.1.1 Kontextualizace

Kontextualizace je proces prvotního nastavení virtuálního stroje. Jedná se o jakousi jednoduchou konfiguraci, která je široce podporovaná mezi poskytovateli cloudové infrastruktury. Pro její využití je potřeba podpora v cloudové platformě, tj. na úrovni virtualizace. V EGI Federated Cloud se využívá na úrovni OCCI standardu cloud-init, což je soubor ve formátu YAML, který může vypadat následovně:

```
#cloud-config
users:
  - name: cloud-user
    sudo: ALL=(ALL) NOPASSWD:ALL
    lock-passwd: true
    ssh-import-id: cloud-user
    ssh-authorized-keys:
      - ssh-rsa MIIBCgKCAQEA+xGZ/
        wcz9ugFpP07Nspo6U17l0YhFiFpxxU4pTk3Lifz9R3zs
        IsuERwta7+fWIfxOo208ett/
        jhskiVodSEt3QBGh4XBipyWopKwZ93HHaDVZAALi/2A+
        xTBtWdEo7XGUujKDvC2/aZKukfjpOiUI8AhLAfjmlcD/
        UZ1QPh0mHsglRNCmpCwmwSXA9V Nmhz+PiB+Dml4WWnKW
        /VHo2ujTXxq7+
        efMU4H2fny3Se3KYOsFPFGZ1TNQSYlFuShWrHPtiL
        mUdPoP6CV2mML1tk+
        l7DIIqXrQhLUKDACeM5roMx0kLhUWB8P+0
        uj1CNlNN4JRZlC7xFfqiMbFRU9Z4N6YwIDAQAB cloud-
        user
```

Kontextualizace je extrémně důležitá, protože se v ní tvoří nastavení potřebné k přihlášení. V daném příkladu, který je využitelný pro stroje na bázi UNIXu, tj. Linux, OpenBSD apod. můžeme vidět několik nastavení:

- Hlavním uživatelem je `cloud-user` a je nastavený jako uživatel s právy root[2]. Důvodem, proč se nepoužívá přímo root je to, že je to bezpečnostním rizikem.
- `lock-passwd` - `passwd` je nástroj pro změnu hesla. Toto nastavení znemožňuje změnu, případě prvotní nastavení hesla.
- `ssh-authorized-keys` je položka, která umožňuje zadaným uživatelům s konkrétním veřejným klíčem přístup k tomuto stroji. Často se používá v cloudových infrastrukturách pro automatizaci přístupu administrátorů.

[2]Tzv. superuživatel v prostředí UNIX-like OS, má plné práva ovládat OS

6.1.2 Create

V této funkci se vytváří samotný virtuální stroj. Parametry, které je třeba předat v konfiguraci jsou následující:

- `endpoint` - Federated Cloud OCCI endpoint
- `x509` - x509 VOMS certifikát pro autentizaci
- `image_template` - obraz virtuálního stroje
- `resource_template` - definuje vlastnosti virtuálního stroje, počet CPU, operační paměti apod.
- `name` - název stroje
- `init_file` - Inicializační soubor pro kontextualizaci
- `storage_size` - velikost úložiště, které má být dostupné
- `network` - síť, ke které se má stroj připojit

Tyto parametry se načítají z konfiguračního souboru ve formátu HCL. Takový soubor může vypadat následovně:

```
resource "occi_virtual_machine" "master" {
   image_template = "http://occi.carach5.ics.muni.cz/occi/infrastructure/
       os_tpl#
       uuid_fe71524e_66d3_5d09_8375_c5510ed5ccba_warg_default_shared_230"
   resource_template = "http://fedcloud.egi.eu/occi/compute/flavour/1.0#
       large"
   endpoint = "https://carach5.ics.muni.cz:11443"
   name = "vm_cluster_master"
   x509 = "/tmp/x509up_u0"
   init_file = "/metapipe-files/mmg-cluster-setup-CESNET/context"
   storage_size = 300
}
resource "occi_virtual_machine" "node" {
   image_template = "http://occi.carach5.ics.muni.cz/occi/infrastructure/
       os_tpl#
       uuid_fe71524e_66d3_5d09_8375_c5510ed5ccba_warg_default_shared_230"
   resource_template = "http://fedcloud.egi.eu/occi/compute/flavour/1.0#
       large"
   endpoint = "https://carach5.ics.muni.cz:11443"
   name = "vm_cluster_node"
   x509 = "/tmp/x509up_u0"
   init_file = "/metapipe-files/mmg-cluster-setup-CESNET/context"
   count = 4
   storage_size = 50
}
```

Tato konfigurace vytváří jeden hlavní mater uzel s úložištěm 300 GB a 4 worker uzly s úložištěm 50 GB. Jako resource template se využije tzv. large flavour, což je kombinace 4 jader CPU a 16 GB RAM operační paměti.

Pro výstup, který následně využije Ansible je třeba ještě si v konfiguraci definovat výstupní proměnné:

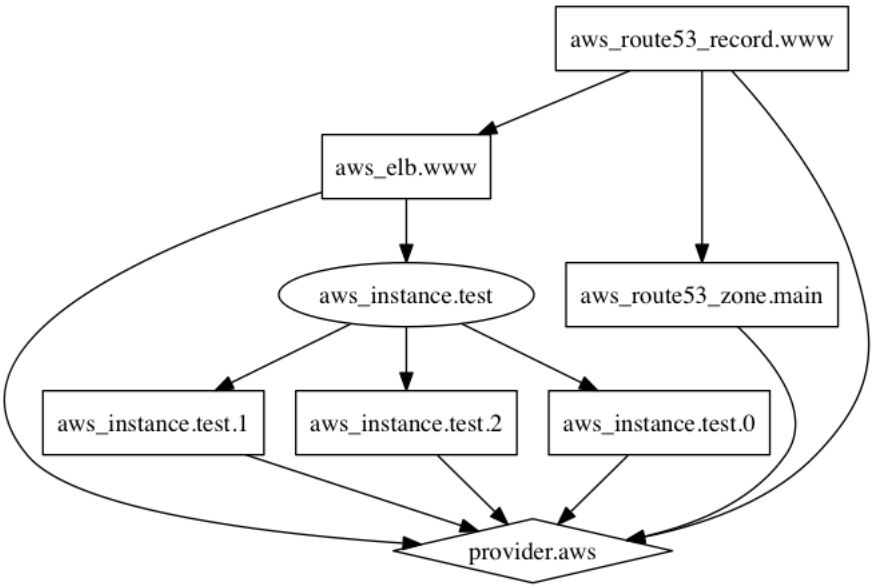
```
output "master_ip" {
   value = "${occi_virtual_machine.master.ip_address}"
}

output "master_id" {
   value = "${occi_virtual_machine.master.id}"
}

output "master_storage_link" {
   value = "${occi_virtual_machine.master.storage_link}"
}

output "node_ip" {
   value = "${join(",",occi_virtual_machine.node.*.ip_address)}"
}

output "node_id" {
   value = "${occi_virtual_machine.node.0.id}"
}

output "master_storage_size" {
   value = "${occi_virtual_machine.master.storage_size}"
}

output "node_storage_size" {
   value = "${occi_virtual_machine.node.0.storage_size}"
}

output "occi_endpoint" {
   value = "${occi_virtual_machine.master.endpoint}"
}

output "proxy_file" {
   value = "${occi_virtual_machine.master.x509}"
}
```

Mezi výstupné proměnné patří:

- `master_ip` - IP adresa master uzlu
- `master_id` - ID master uzlu. Tím se stroj identifikuje v rámci OCCI prostředí.
- `master_storage_link` - Link na připojené úložiště master uzlu
- `node_ip` - Seznam IP adres worker uzlů
- `node_id` - Seznam ID worker uzlů
- `master_storage_size` - Velikost úložiště master uzlu v GB
- `node_storage_size` Velikost úložiště jednotlivých worker uzlů v GB

Jelikož v OCCI je úložiště a samotný výpočetní stroj chápany jako dva různé prvky, je třeba při tvorbě úložiště je prvně vytvořit podobným postupem jako virtuální stroj a po vytvoření virtuálního stroje toto úložiště ke stroji připojit.

Po načtení všech promměných se zavolá rOCCI-cli se zadanými parametry. Návratová hodnota tohoto příkazu je ID právě vytvořeného stroje. Terraform funguje deklarativně a paralelně, což znamená, že jednotlivé instance tvoří podle toho, jak jsou na to závislé a pokud je to možné, tak je tvoří zároveň. Tzn. že se prvně musí vytvořit úložiště a až pak virtuální stroj, ke kterému se připojí.



Obr. 6.1: Závislostní graf pro vytvoření infrastruktury [13]

6.1.3 Read

Funkce read čte proměnné z konfiguračního souboru. Její výstup může být jak ve strojově čitelném formátu JSON, tak ve formátu čitelném pro člověka.

6.1.4 Update

Funkce update při změně konfiguračního souboru provede patřičné změny na infrastruktuře. Jelikož je infrastruktura principiálně immutable, při změně se patřičná instance zničí a vytvoří se nová s patřičnými parametry.

6.1.5 Delete

Funkce delete zničí příslušný virtuální stroj a s ní i úložiště k němu připojené. To se provede v tomhle pořadí:

1. Úložiště se odlinkne od virtuálního stroje
2. Zničí se úložiště
3. Zničí se virtuální stroj

6.2 Ansible playbooks

Z původního MMG Cluster setup nástroje byly adaptovány Ansible playbooky, nicméně je třeba je značně upravit. Musí se odstranit nepotřebná funkcionalita (např. Hortonworks, Ambari, instalace Bastionu apod.), stejně se musí vypustit původní orchestrace, která byla navržena na OpenStackovou infrastrukturu cPouta.

Jako vstup z předchozího kroku je soubor inventáře ve formátu INI, který byl vytvořen jednoduchým BASH skriptem z výstupu Terraformu. Tento soubor může vypadat následovně:

```
[all:vars]
ansible_ssh_user=cloud-user
ansible_sudo=true
cluster_name=csc-cluster
[masters]
csc-cluster-master ansible_ssh_host=147.228.242.187 vm_group_name=master
[nodes]
csc-cluster-node-1 ansible_ssh_host=147.228.242.183 vm_group_name=disk
csc-cluster-node-2 ansible_ssh_host=147.228.242.184 vm_group_name=disk
csc-cluster-node-3 ansible_ssh_host=147.228.242.185 vm_group_name=disk
csc-cluster-node-4 ansible_ssh_host=147.228.242.186 vm_group_name=disk
```

V souboru lze vidět nastavení globálních proměnných ve skupině `[all:vars]`. Proměnná `ansible_ssh_user` říká Ansible, že má pro SSH přístup využít uživatele cloud-user místo standardního `root` uživatele a tento uživatel má zapnuté využití `root` privilegií. Následně jsou rozřazené jednotlivé stroje do skupin master a worker nodes.

6.2.1 Role

V této podkapitole jsou popsány jednotlivé role v playboocích.

Base

Tato role je aplikovaná pro všechny stroje. Provádí tyto kroky:

1. Instalace základních nástrojů - instaluje se na čistý operační systém, v našem případě CentOS, kde spoustu nástrojů chybí
   - `dstat`
   - `lsof`
   - `bash-completion`
   - `time`
   - `tmux`
   - `git`
   - `xauth`
   - `screen`
   - `nano`
   - `vim`
   - `bind-utils`
   - `nmap-ncat`
   - `lvm2`
   - `chrony`
   - `bzip2`
   - `iptables-services`
   - `perl-CPAN`
   - `perl-Digest-MD5`
   - `sudo`
   - `wget`
2. Úprava `/etc/hosts/`[3] souboru pro IPv4 adresy
3. Úprava `/etc/hosts/` souboru pro IPv6 adresy
4. Odstranění ntp.org serverů z konfigurace
5. Přidání vlastních NTP[4] serverů
6. Odstranění `firewalld` nástroje
7. Zapnutí `iptables`[5]

[3]Ekvivalent Hosts souboru na Windows, tento soubor definuje pevně dané mapování hostnames na IP adresy, např. localhost na 127.0.0.1

[4]Network time protocol je protokol pro synchronizaci času počítačů po síti

[5]Nástroj pro nastavování pravidel firewallu v jádře Linuxu

Cluster common

Tato role byla původně aplikována na všechny stroje v clusteru, tzn. mimo Bastion hosta, ale vzhledem k vypuštění Bastion hosta je aplikována stejně jako role Base. Tato role provádí kroky:

1. Namíření `resolv.conf` na ten na master uzlu
2. Vytvoření `dhclient.conf` souboru
3. Konfigurace DNS tak, aby se využil DNS server na master uzlu přes soubor `dhclient.conf` [6]
4. Přidání perzistentního `iptables` souboru
5. Generování SSH klíče pro uživatele `cloud-user`
6. Překopírování SSH klíče na worker uzly
7. Autorizace SSH klíčem
8. Nastavení SELinuxu [7]

Cluster master

Toto je role pouze pro master uzel. Zde jsou provedeny tyto kroky:

1. Přidání záznamů všech worker uzlů do `/etc/hosts` souboru
2. Instalace `pdsh` balíčku
3. Otevření portů ve firewallu pro IP adresy worker uzlů
4. Otevření portu 8080 do internetu pro webové UI Sparku
5. Otevření portu 4040 do internetu pro META-pipe UI
6. Otevření portu 7077 do internetu pro Spark API
7. Otevření portu 6066 pro REST API
8. Restartování `iptables`

Cluster node

Tato role je pro změnu aplikována pouze pro worker uzly.

1. Přidání záznamů všech uzlů do `/etc/hosts` souboru
2. Otevření portů ve firewallu pro IP adresu master uzlu
3. Otevření portů ve firewallu pro IP adresy worker uzlů
4. Otevření portu 8081 pro webové UI
5. Restartování `iptables`

[6] Konfigurační soubor DHCP klienta

[7] Rozšíření jádra Linuxu o povinné řízení přístupu. Slouží ke zvýšení bezpečnosti

LVM storage

Logical volume management je metoda správy diskového prostoru používaná v Linuxu. Ta je využita též v Ansible pro správu úložného prostoru virtuálních strojů:

1. Zjistí se, zda-li už nebylo naformátované úložiště. Pokud ne, pokračuje se dále
2. Aktivace logických svazků
3. Vytvoření fyzického svazku
4. Vytvoření skupiny svazků
5. Vytvoření logického svazku na fyzickém
6. Formátování svazku
7. Připojení svazku
8. Kontrola, zda-li swap existuje
9. Vytvoření swapu na logickém svazku
10. Připojení swapu
11. Přidání záznamů úložiště a swapu do souboru `/etc/fstab/`[8]

NFS server

Tato role je určená pro master uzel, ve které se konfiguruje jeho role jako NFS hosta.

1. Instalace NFS balíčku `nfs-utils`
2. Instalace NFS server balíčku `nfs-server` a `rpcbind`[9]
3. Aktivate `rpcbind`
4. Vytvoření složky, která se bude sdílet a nastavení sdílecích práv

NFS client

V této roli se konfigurují worker uzly, aby používaly NFS úložiště připojené k master uzlu.

1. Instalace NFS balíčku `nfs-utils`
2. Ověření, zda-li existuje sdílená složka a jestli se k ní lze připojit
3. Připojení sdílení složky přes `fstab`

[8] fstab je soubor, který v UNIX-like operačních systémech definuje seznam všech připojených diskových oddílů a jejich umístění v souborovém systému

[9] rpcbind mapuje RPC služby na porty, na kterých naslouchají. Využívá se především pro NFS

6.3 MMG Cluster Setup

MMG Cluster se skládá ze souboru několika Python skriptů:

- `create.py`
- `run.py`
- `stop.py`
- `destroy.py`

Každý z těchto skriptů tvoří jednu z funkcionalit nástroje. Pro jeho funkčnost je třeba zaručit přítomnost nástrojů na klientovi:

- `rOCCI-cli` balíček
- X509 VOMS certifikát
- Kontextualizační soubor
- Terraform s OCCI pluginem
- Ansible

6.3.1 create.py

Tento skript obaluje Terraform a Ansible a je nejdůležitější částí programu. Postupně volá nástroje v určitém pořadí a skládá tak infratrukturu:

1. Orchestruje infrastrukturu příkazem `terraform apply`
2. Pomocí příkazu `terraform show` zjistí potřebné proměnné pro vytvoření souboru inventáře pro Ansible
3. Vytvoří seznam IP adres worker uzlů
4. Zjistí jednotlivé parametry infrastruktury. To nelze vyčíst přímo z Terraformu, ale je třeba postupně volat rOCCI-cli na každý parametr. Tyto parametry jsou:
   - Počet CPU jader na master uzlu
   - Množství RAM na master uzlu
   - Velikost úložiště na master uzlu
   - Počet CPU jader na worker uzlech
   - Množství RAM na worker uzlech
   - Velikost úložiště na worker uzlech
5. Ze zjištěných parametrů se vygeneruje soubor `cluster_vars.yaml`, který se následně předá Ansible pro konfiguraci
6. Vytvoří se konfigurace pro Spark hosta z dostupného množství CPU jader a RAM
7. Překopírují se potřebné soubory na jednotlivé stroje
8. Zavolá se Ansible
9. Pokud byl při spuštění předán parametr ke spuštění clusteru, tak se spustí cluster

Vygenerovaný soubor `cluster_vars.yaml` může vypadat takto:

```
master:
    inventory_group: masters
    auto_ip: yes
    flavor: "1.0#small"
    volumes:
    - name: metadata
        size: 200
        pv_path: /dev/vdc
    filesystems:
    - name: swap
        volume: metadata
        size: "2%VG"
        fstype: swap
    - name: hadoop
        volume: metadata
        mount_path: /hadoop
        size: "97%VG"
        fstype: xfs
        mkfs_opts: ""
node:
    flavor: "1.0#small"
    num_vms: 4
    volumes:
    - name: datavol
        size: 100
        pv_path: "/dev/vdc"
    filesystems:
    - name: swap
        volume: datavol
        size: "2%VG"
        fstype: swap
    - name: hadoop_disk
        volume: datavol
        size: "97%VG"
        mount_path: /hadoop/disk
        fstype: xfs
```

6.3.2 run.py

Tento skript obsahuje kód pro spouštění samotného clusteru a je možné jej volat přímo z `create.py` při zadání parametru při spuštění. Je vyžadován parametr `job-tag`, který říká clusteru, které úlohy si má brát z centrálního serveru METApipe.

Kontroluje se, zda-li existuje soubor inventáře Ansible, aby se nestalo, že by se skript spustil na neexistující cluster. Následně se z inventáře vezme IP adresa master uzlu a následně se zavolá jednoduchý BASH skript, který zapne cluster v módu funkcionální analýzy. Pro potřeby téhle aplikace nebyla zatím implementovaná integrace assembly módu. Po spuštění clusteru se vypíše IP adresa a port směřující na webové UI Sparku.

6.3.3 stop.py

Tento jednoduchý skript zastavuje běžící cluster. Volá se ze skriptu `destroy.py`, než se zničí cluster.

6.3.4 destroy.py

Tento skript prvně zastaví cluster skriptem `stop.py` a následně zničí a uklidí infrastrukturu příkazem `terraform destroy`. Následně smaže soubory, které byly vytvořené v předchozích krocích jako `terraform.tfstate`, `cluster_vars.yaml` či `ansible_inventory`.

7 OPTIMALIZACE

V předchozí kapitole jsme implementovali META-pipe v prostředí EGI Federated Cloud. Nicméně tato implementace má několik nedostatků z původního návrhu:

- Omezení výkonu kvůli nízkému IO výkonu NFS, obzvláště na virtuálních strojích s připojeným klasickým plotnovým pevným diskem místo SSD
- Velmi komplikovaná instalace

V této kapitole se bude práce věnovat řešení, které tyto problémy zredukují.

7.1 CernVM File System

CernVM File System (CernVM-FS) poskytuje škálovatelnou, spolehlivou a nízkoúdržbovou službu pro distribuci software. Byla vyvinuta pro asistenci High Energy Physics (HEP) kolaborací pro nasazení software na celosvětově distribuované výpočetní infrastruktuře používanou k běhu aplikací zpracovávající data. CernVM-FS je implementovaný jako POSIX read-only file system - FUSE[1] modul. Soubory a složky jsou hostované na standardních webových serverech a mountované v univerzáním namespace `/cvmfs`.

Interně CernVM-FS využívá obsahově adresovatelné úložiště a Merkleovy stromy[2] ke správě dat a metadat. CernVM-FS užívá pouze odchozí HTTP připojení, čímž se vyhýbá většině problémů souvisejících s firewallem jiných síťových protokolů jako NFS. Přenáší data a metadata on-demand a verifikuje integraci dat kryptografickým hašem.

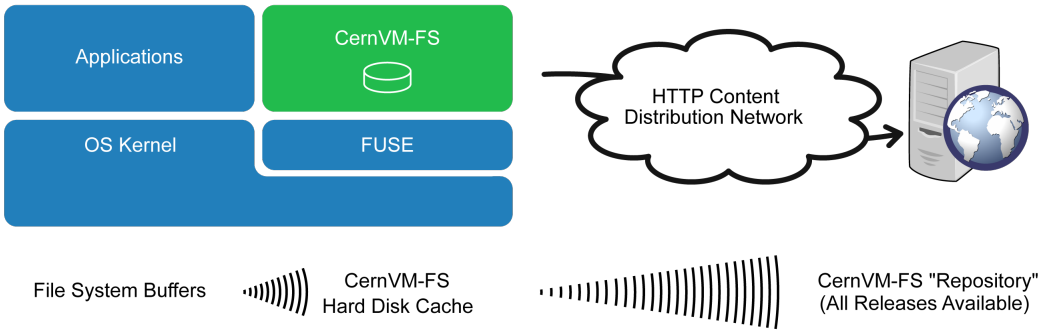
CernVM-FS se zaměřuje specificky na softwarové use case pomocí agresivního cachingu[3] a redukcí odezvy. Software je typicky složen z mnoha malých souborů, které jsou často otevírány a čteny jako celek. Jeden z use case obsahuje časté vyhledávání souborů ve vícero složkách, kde jsou ověřovány vyhledávací cesty.

CernVM-FS je aktivně využíván menšími i většími HEP projekty. V mnoha případech nahrazuje balíčkovací managery a sdílené softwarová úložiště na clusterových file systémech jako nástroj pro distribuci softwaru použitého k zpracování experimentálních dat. Pro experimenty v LHC CernVM-FS hostuje několik stovek milionů souborů a složek, která jsou distribuována ke stovkám tisícům počítačů.

[1] Filesystem in UserSpacE je rozhrání pro userspace programy k exportu filesystemu Linux jádru

[2] Merkleův, též známý jako hašový strom je datová struktura stromu, která má v listech data a v ostatních vrcholech výsledky kryptografické hašovací funkce. Jeho výhoda spočívá v tom, že lze ověřit jeho integritu v logaritmickém čase

[3] Cache, též mezipaměť je hardwarová nebo softwarová součást počítače, která uchovává data pro rychlejší přístup k nim



Obr. 7.1: Architektura CernVM-FS [14]

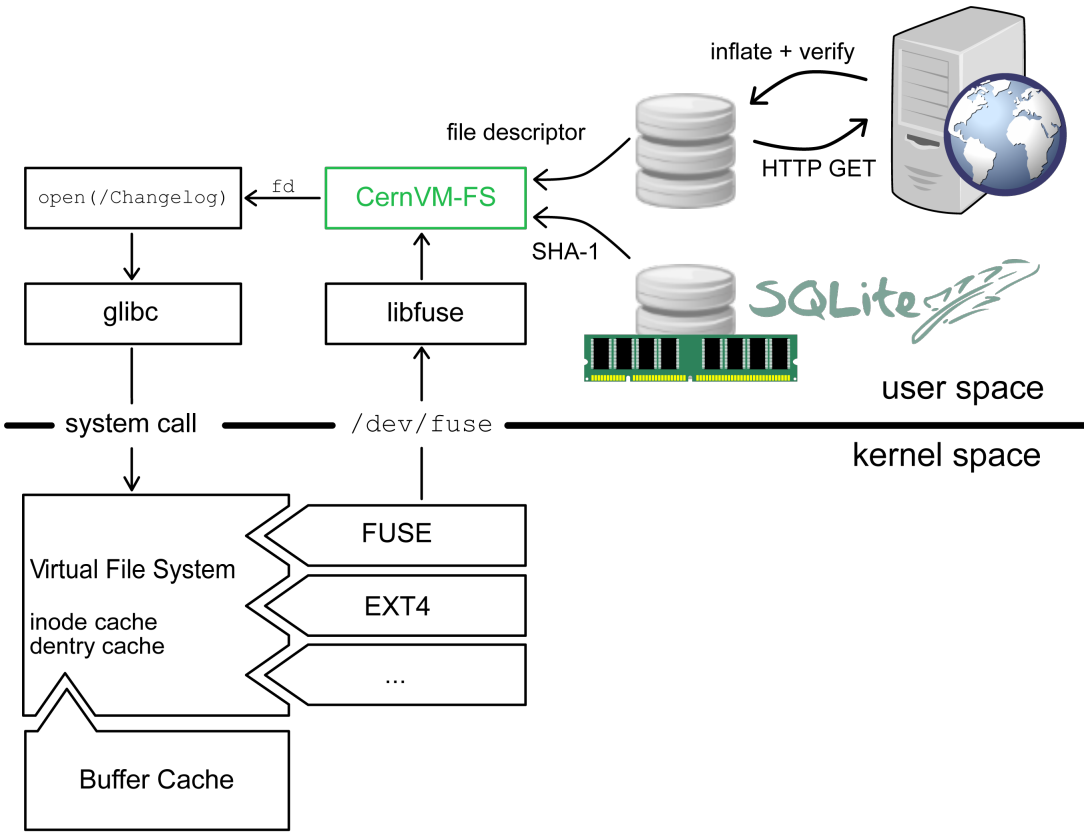
Současná implementace CernVM-FS nabízí tyto funkcionality:

- Management cache quota
- Použití FUSE kernel modulu s in-kernel cachingem atributů souborů
- Použití obsahově adresovatelného formátu úložiště vedoucí k immutable souborů a automatické deduplikaci
- Možnost rozdělit hiearchii složek na podkatalogy
- Automatické aktualizace katalogů souborů kontrolované TTL[4] uloženém uvnitř katalogů
- Digitálně podepsané repozitáře
- Transparentní souborová komprese/dekomprese a transparentní file chunking
- Možnost pracovat v offline módu za předpokladu že všechny potřebné soubory jsou cachované
- Verzování souborů
- Hotpatching klienta filesystemu
- Dynamická expanze environmental variables v symbolických lincích
- Podpora pro rozšířené atributy, např. atributy SELinuxu
- Automatický mirroring serverů na bázi geografického umístění
- Automatický load balancing proxy serverů
- Efektivní replikace repozitářů
- Možnost použití S3[5] kompatibilního úložiště místo interního filesystemu jako repozitář

[4]Time to live definuje dobu, po kterou mají být data zachovány, než budou zahozena
[5]Amazon S3 je komerční služba pro ukládání dat

Narozdíl od všeobecných protokolů pro filesystémy jako NFS nebo AFS je CernVM-FS vyvinutý přímo pro rychlé a škálovatelné softwarové distribuace. Běh a kompilace softwaru je use case, na který není všeobecný protokol jako NFS stavěn. Narozdíl od obrazů virtuálních strojů nebo obrazů Dockeru není třeba software instalovaný v CernVM-FS dále balíčkovat. Místo toho je distribuován a verzován soubor po souboru. Pro vytvoření a aktualizaci CernVM-FS repozitáře je třeba specifický stroj zvaný Release Manager Machine. Na tom je CernVM-FS repozitář mountovaný v módu čtení/zápis způsobem union file system[6]. Union file system je nadřazený CernVM-FS read-only pomocí zapisovatelného oddílu úložiště. CernVM-FS server toolkit následně spojí změny zapsané do zapisovatelného oddílu. Spojení a publikace změn může být spuštěna automaticky v čase. Jedná se o atomickou operaci, které musí proběhnout buď celá, nebo vůbec. CernVM-FS repozitář se tak podobá repozitáři verzovacího systému Git.

Na klientovi, který používá data z CernVM-FS repozitáře se stahují a cachují pouze data a metadata software, které jsou potřeba pro spuštění.

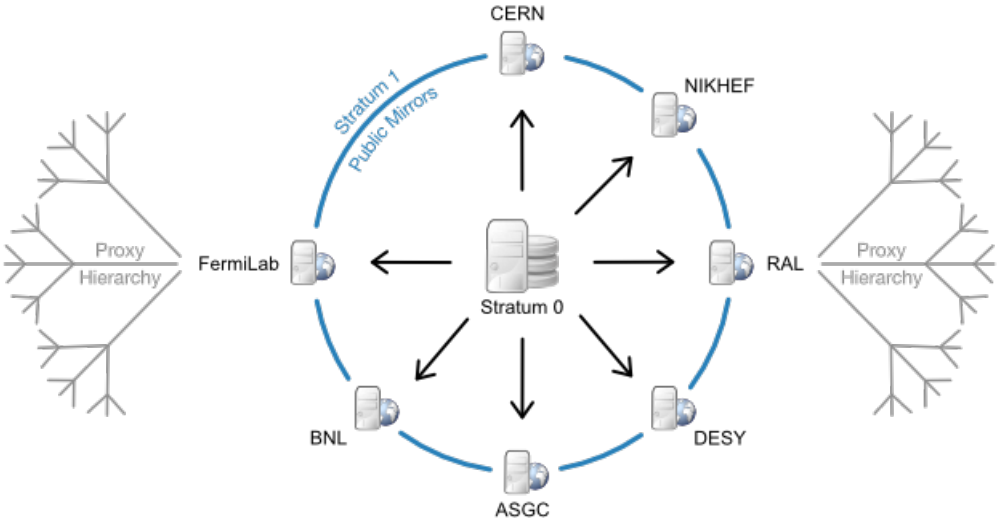


Obr. 7.2: Synchronizace CernVM-FS [14]

[6]Implementované pomocí služby Unionfs umožňuje souborů a složkám separátních filesystemů transparentní překrytí

CernVM-FS byl vyvinut CERNem pro distribuci velkého množství software velkému množství klientů. To se hodí právě META-pipe, která vyžaduje databáze, které jsou principiálně kolekcí velkého množství souborů.

V rámci této práce CernVM-FS bude použit pro hostování databáze a software pro běh META-pipe.



Obr. 7.3: Centrální Stratum 0 repozitář CernVM-FS a replikační servery [14]

7.2 Squid

Squid je caching proxy pro Web podporující HTTP, HTTPS FTP a další protokolu. Redukuje přenos dat a zlepšuje čas odevzdy cachováním a znovupoužitím často vyžadovaných webových stránek. Squid má rozsáhlé možnosti nastavení přístupu a běží na většině dostupných operačních systémech včetně Windows a je licencován pod licencí GNU GPL. Squid běží jako daemon[7] na UNIX-like systémech. Na Windows běží momentálně pod prostředím Cygwin[8].

Po instalaci Squid proxy serveru mohou webové prohlížeče být nakonfigurovány tak, aby Squid používali jako HTTP proxy server, umožňující Squidu ponechávat si kopie otevřených dokumentů, což při opakovaných požadavcích po stejném dokumentu redukuje přístupový čas stejně jako přenesená data. To je často užitečné pro poskytovatele Internetu k zvýšení rychlosti jejich zákazníkům a lokálním sítím sdílejících internetové připojení.

[7]Daemon je program který běží jako proces na pozadí, narozdíl od typického programu, který je plně v režii uživatele

[8]POSIX kompatibilní prostředí běžící na Windows umožňující běh programům psaných na UNIX-like operačních systémech

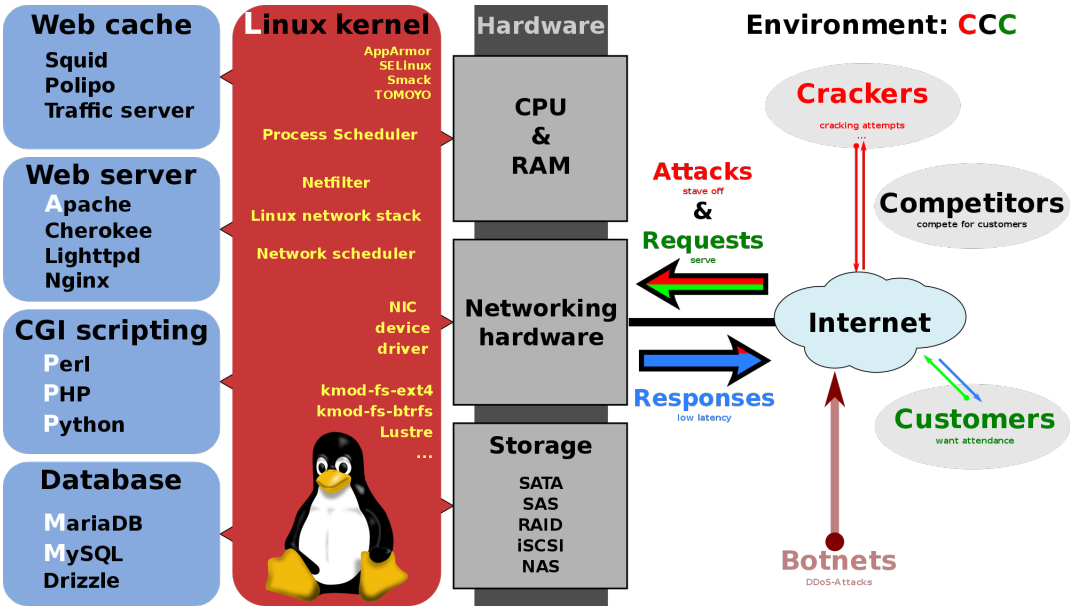
Klient, typicky webový prohlížeč může buď explicitně specifikovat proy server který chce využívat (typické pro zákazníky poskytovatele internetu) nebo může používat proxy bez dalšího nastavení. Transparentní caching je přístup, ve kterém všechny odchozí HTTP požadavky jsou zachycené Squidem a odpovědi jsou cachované. Tento přístup je typicky využit v korporátním prostředí, kde jsou všichni klienti na stejné lokální síti. Toto řešení ale přináší problémy s osobními daty.

Squid obsahuje funkce, které pomáhají anonymizaci připojení, např. vypnutí či změna specifických hlaviček v klientských HTTP požadavcích. Lidé surfující na síti s transparentním cachováním Squidu nemusí vědět o tom, zda-li tato informace je zaznamenávána.

Squid může též fungovat jako reverzní proxy. Ve výše uvedeném případu klasické proxy se využívá cachování neomezeného množství webových serverů pro limitované množství klientů. V reverzní proxy cache slouží neomezenému množství klientů pro limitované množství webserverů, popř. jen pro jeden. Existuje možnost pro jediný Squid server fungovat jak proxy, tak reverzní proxy.

Vzhledem k limitaci běžného HTTP protokolu je Squid omezen cachováním jistých požadavků. To se částečně eliminuje podporou částečných požadavků, např. na videoportálu YouTube.

Pro META-pipe má Squid využití jako cachovací server pro CernVM-FS, který běží na HTTP protokolu.



Obr. 7.4: Squid jako součást LAMP stacku [15]

7.3 Úprava návrhu

Do původního návrhu bude integrován CernVM-FS a Squid. To vyžaduje úpravu architektury, kdy se NFS nahradí touto kombinací. Tím se značně urychlí servírování dat v rámci clusteru.

7.3.1 Ansible role

Pro Ansible je třeba vytvořit dvě nové role, jednu pro Squid hosta provozovaném na master uzlu a jednu pro CernVM-FS klienta, který budou používat všechny uzly.

Squid server

Zde je potřeba udělat tyhle kroky v tomhle pořadí:

1. Instalace `squid` balíčku
2. Vytvoření squid cache složky
3. Vytvoření symbolického linku do cache složky
4. Editace squid parametrů v `squid.conf` souboru jako je `max_filedesc`, `maximum_object_size`, `cache_mem` a `maximum_object_size_in_memory`
5. Nastavení cache složky
6. Přidání CernVM-FS repozitáře do Squid ACL[9]
7. Přidání povolených IP pro přístup ke Squidu
8. Povolení HTTP přístupu pro CernVM-FS
9. Zapnutí Squidu

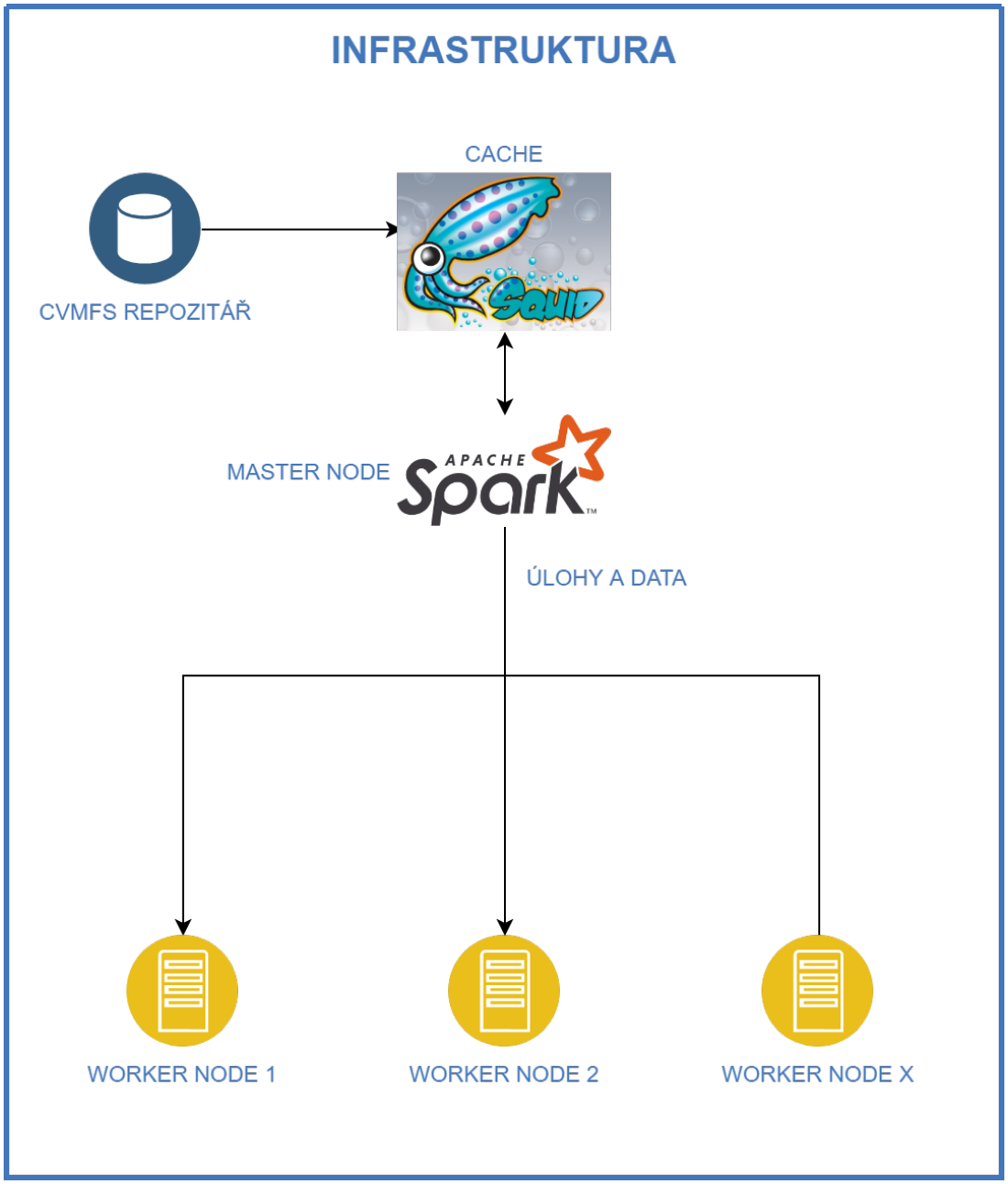
CernVM-FS client

CernVM-FS klient je třeba nainstalovat na všechny uzly v clusteru provedením:

1. Přidání `cvmfs` repozitáře
2. Instalace `cvmfs` a `cvmfs-config-default` balíčku
3. Nastavení základní konfigurace
4. Přidání META-pipe repozitáře
5. Přidání lokální Squid proxy běžící na master uzlu
6. Stažení veřejného klíče CernVM-FS repozitáře
7. Přidání URL repozitáře
8. Přidání veřejného klíče do repozitáře
9. Mount repozitáře v filesystému

[9] Access control list je seznam oprávnění připojený k nějakému objektu, který definuje kdo nebo co má jaké povolení

7.3.2 Upravená architektura



Obr. 7.5: Nová architektura infrastruktury MMG Cluster setup. Zdroj: autor

V nové architektuře jednotlivé uzly přistupují k databázím proteinům a METApipe softwaru přes cachovací proxy Squid, která si prvky bere z vzdáleného CernVMFS repozitáře. Tento repozitář může být umístěn kdekoliv na světě. To značně zrychluje přístup k databázím vzhledem k tomu, že Squid pro caching používá operační paměť a nespoléhá se tak na rychlost obvykle pomalých plotnových pevných disků.

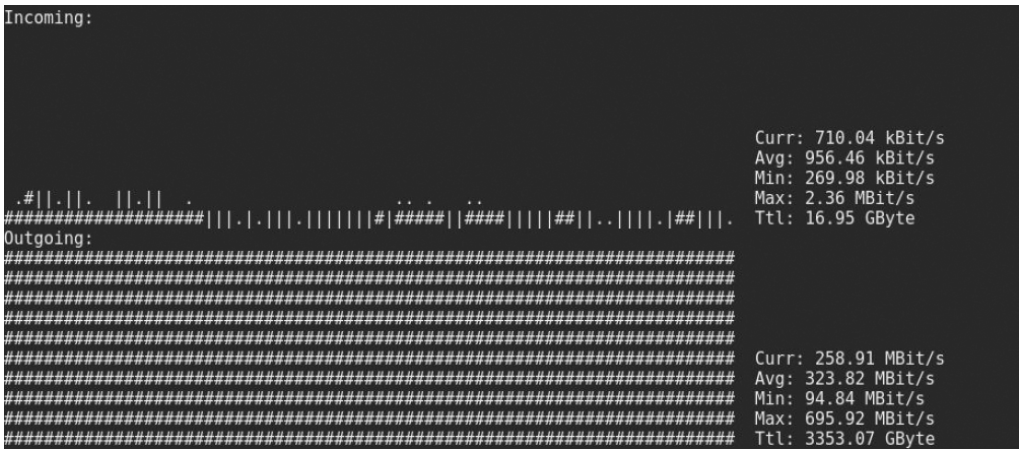
7.4 Benchmarking

Po upravení návrhu došlo k značném zlepšení doby výpočtu funkcionální analýzy díky téměř neomezenémi IO mezi master uzlem a worker uzly. Typické plotnové disky při čtení malých souborů mají rychlost v řádech MB/s. Při využití Squid cache je možné využít plné linky serveru (typicky 1 Gb/s, u novějších 10 Gb/s) vzhledem k rychlosti operační paměti (řádově desítky GB/s u běžných DDR3 pamětí). Takto je možné přenášet data po síti řádově rychleji.

Pro benchmarking byly využity vzorky běžných sekvencí proteinů, který může vypadat následovně, jedná se o formát souboru FASTA[10] lze v něm vidět contig a jednotlivé proteiny:

```
>contig-25000016 contig-25010016 undefined product 7778394
:7779646 forward
ggttaaagtgtaccaatataacgacgatagcttaatgttacacaatgatttatatcaaat
taatatggctgaaagctactggaatgatggtatccatgaaagaatagcagtgtttgattt
gtattttcgaaaaatgccatttaatagtggatatgcggtattcaacggattgaaacgcgt
tgtgaatttcatcgaaaactttgggtttacaaatgaagatatcacatatttaaaatcgat
aggttacgaagaagattttctaaactacctaaaagatttgaaatttacagggaatattaa
atctatgcaagaaggtgaaatttgttttggtaatgagccattattaagagttgaagcacc
tttaatccaagcgcaacttattgaaactattttgttaaatatcattaatttccaaacatt
aattgcaactaaagccagccgaattcgtcaaatagcaacgcatgacactttgatggaatt
tggtacaagaagagctcaagagatcgatgctgcactgtggggcgctagagcagcctttat
tggagggtttgattctacaagtaatgttagagcaggaaaacttttaatatacctgtatc
tggcacacatgcacacgcactagtacaaacatatggtgatgagtatatagcattcaaaaa
gtatgctgagcgacataaaaattgtgtgttcttagttgatacttttcatactttaaaatc
aggagtaccaaccgcaattaaggttgcaaaagagttaggagatactattaattttatagg
tatcagattagattctggtgatattgcgtacctatctaaagaagctcgtagaatgttaga
tgaggctggttttacagaagctaaaattatcgcatcaaatgatttggatgagcagactat
tacaagtttaaaagcacaaggcgctaaagttgacggatggggagtaggtacaaaactgat
tacaggatatgatcaaccagccttaggtgcagtttataaattggtttctattgaaacaga
tgatggcacaatgagtgatcgcattaaattatcaaataatgctgagaaagttactacacc
aggcaaaaaaaatgtttatcgtattattaataataaaacaggcaaggctgagggcgacta
tattacgctagaaggtgaaaatcctaatgacgaatctccattgaaaatgttccatcctgt
tcacacttacaaaatgaagtttattaaatcatttaaagcggttaatctacatc
```

[10]Univerzální standard v bioinformatice a biochemii, textový formát reprezentující sekvence DNA nebo sekvence aminokyselin proteinů, které jsou reprezentovány jednopísmenným kódem. Pochází ze stejnojmenného softwarového balíčku



Obr. 7.6: Běžný síťový provoz master uzlu, Zdroj: autor

Benchmark byl proveden v módu funkcionální analýzy na každém vzorku 5x a z výsledků se udělal aritmetický průměr. Výsledky byly naměřeny následovně:

Vzorek	Čas před úpravou	Čas po úpravě	Zlepšení
Malý	89 s	49 s	82%
Střední	156 s	91 s	71%
Velký	373 s	244 s	52%

Tab. 7.1: Porovnání času výpočtu před a po úpravě architektury, Zdroj: autor

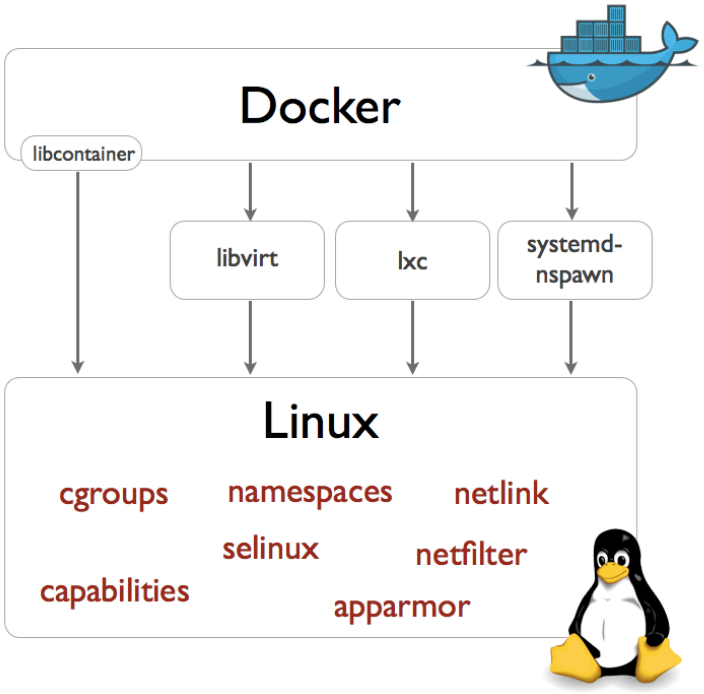
Po úpravě architektury došlo k výraznému zlepšení doby analýzy u jednotlivých vzorků, které reprezentují skutečné využití META-pipe.

7.5 Docker

Docker je počítačový program provádějící virtualizaci na úrovni operačního systému. To se využívá k běhu softwarových balíčků zvaných kontejnery. Kontejnery jsou izolované od sebe a obsahují vlastní aplikaci, nástroje, knihovny a konfigurační soubory. Pro komunikaci mezi sebou používají speciální rozhrání. Všechny kontejnery běží na jediném jádře operačního systému a tím pádem mají menší režii jako obvyklé virtuální stroje. Kontejnery jsou vytvářeny z obrazů podobně jako virtuální stroje. Obrazy specifikují jejich přesný obsah a často jsou vytvářeny kombinováním a modifikací standardních obrazů dostupných z veřejných repozitářů.

Docker je vyvíjený především pro Linux, kde využívá funkcionality izolace zdrojů Linuxového jádra jako `cgroups`, `kernel namespace` a unijní filesystemy jako OverlayFS pro běh nezávislých kontejnerů na jediné Linuxové instanci, čímž se vyhýbá

režii startování a údržbě virtuálních strojů. Podpora Linuxového jádra pro namespaces izoluje aplikační pohled operačního prostředí včetně stromu procesů, sítě, uživatelských ID a mountovaných filesystemů, přičemž `cgroups` jádra poskytuje omezení prostředků pro operační paměť a CPU. Docker též obsahuje `libcontainer` knihovnu jako svoji vlastní cestu, jak přímo využívat virtualizační kapacity Linuxového jádra vedle abstraktních virtualizačních rozhraní přes `libvirt`, LXC a `systemd-nspawn`.



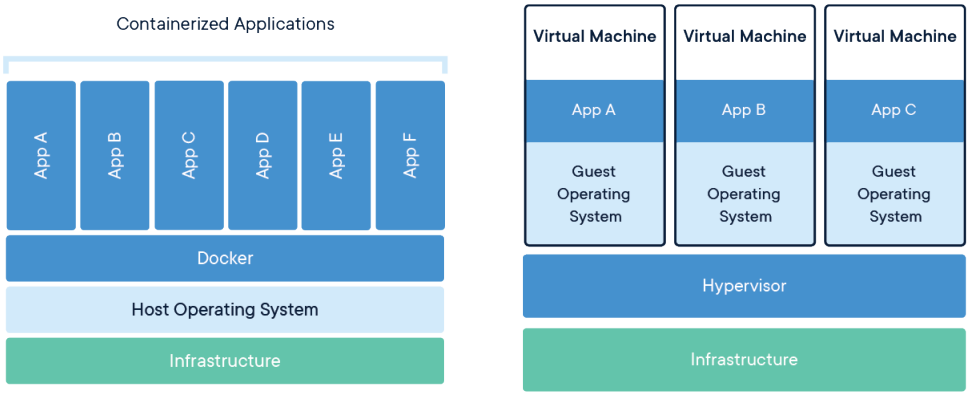
Obr. 7.7: Komunikace Dockeru s Linux jádrem [16]

7.5.1 Komponenty

Docker je služba skládající se ze tří komponent:

- Software - Docker daemon, zvaný `dockerd` je perzistentní proces který řídí Docker kontejnery a obsluhuje objekty kontejneru. Daemon naslouchá na požadavky poslané přes Docker Engine API. Docker klient zvaný `docker` poskytuje rozhraní příkazové řádky, které uživateli umožňuje komunikaci s Docker daemony
- Objekty - Docker objekty jsou různé entity používané ke složení aplikace v Dockeru. Hlavní třídy Docker objektů jsou obrazy, kontejnery a služby:

  - Docker kontejner je standardizované, enkapsulované prostředí které spouští aplikace. Kontejner je řízen pomocí Docker API nebo příkazové řádky
  - Docker obraz je read-only template používaný pro budování kontejnerů. Obrazy jsou používáy k ukládání a nasazení aplikací
  - Docker služba dovoluje kontejnerům škálovat v rámci několika Docker daemonů. Vícero takto spolupracujícím daemonům se říká `swarm`. Komunikují spolu přes Docker API
- Registry - Docker registr je repozitář pro Docker obrazy. Docker klienti se připojují k registrům pro stažení (pull) obrazů pro používání nebo nahrávání (push) obrazů, které postavili. Registry mohou být veřejné nebo soukromé. Mezi dva hlavní veřejné repozitáře patří Docker Hub a Docker Cloud. Docker Hub je implicitní registr kde Docker hledá obrazy. Docker registry také umožňují tvorbu notifikací v závislosti na událostech.



Obr. 7.8: Rozdílná architektura Docker kontejnerů a tradičních virtuálních strojů [16]

Docker implementuje vysokoúrovňové API k běhu lehkých kontejnerů, které spouštějí procesy v izolaci. Díky tomu jeden server nebo virtuální stroj může spustit několik kontejnerů najednou, což bylo díky větší režii virtuálních strojů dříve nemožné. Typický host spouští 5 kontejnerů Dockeru, ale mnoho organizací spouští více jak 10 na silnějších strojích.

Použití kontejnerů zjednodušuje tvorbu vysoce distribuovaných systémů tím, že umožňuje běh vícero aplikací, worker úloh a jiných procesů autonomně na jednom fyzickém stroji nebo více virtuálních strojích. To umožňuje nasazení uzlů podle potřeby ke škálování pro systémy jako Apache Cassandra.

7.5.2 Nástroje

Docker Compose

Docker Compose je nástroj pro definování a běh multi-kontejnerových Docker aplikací. Používá YAML soubory pro konfiguraci aplikačních služeb a provádí vytváření a start procesů všech kontejnerů jedním příkazem. `docker-compose` příkazová řádka dovoluje uživatelům běh příkazů na vícero kontejnerech zároveň, např. budování obrazů, škálování kontejnerů, restart kontejnerů které byly zastaveny apod.

Docker Swarm

Docker Swarm poskytuje nativní clustering funkcionalitu pro Dockerové kontejnery, která mění skupinu Docker enginů v jeden virtuální Docker engine.

7.5.3 Použití v META-pipe

Vzhledem k velkému počtu nástrojů potřebných pro běh MMG Cluster Setup (např. VOMS klient, Python, rOCCI-cli, atd.) je ideální zjednodušit postup instalace. Docker se dá využít nejenom pro běh perzistentních aplikací, ale i pro balíčkování aplikací. Do docker kontejneru jde přidat software verze, která je přesně potřeba a není třeba ověřovat kompatibilitu s softwarem běžícím přímo na stroji uživatele.

Pro nachystání Docker obrazu sou třeba následující soubory:

- Kontextualizační soubor ve formátu `cloud-init`
- `id_rsa` Privátní klíč
- Konfigurační soubor Terraformu `mmg_cluster.tf`
- `usercert.pem` a `userkey.pem` pokud je používaná `fedcloud.egi.eu` základní virtuální organizace
- `elixirx509` certifikát pokud je používaná ELIXIR VO

Docker tvoří obrazy podle `Dockerfile`, který definuje přesný postup, jak se má obraz tvořit krok po kroku. `Dockerfile` pro META-pipe může vypadat následovně:

```
FROM golang:alpine
RUN apk add --update git bash openssh make
WORKDIR $GOPATH/src/github.com/hashicorp/terraform
ENV TF_DEV=true
RUN git clone https://github.com/cduongt/terraform.git
   ./ &&& \
    make dev

FROM centos:6
```

```
WORKDIR /metapipe-install
COPY --from=0 /go/src/github.com/hashicorp/terraform/bin/
   terraform /bin
RUN yum -y install git wget epel-release openssh-clients
RUN yum -y update && yum -y install ansible python34
RUN echo -e "[defaults]\nhost_key_checking␣=␣False" >>
   $HOME/.ansible.cfg
ADD https://raw.githubusercontent.com/EGI-FCTF/fedcloud-
   userinterface/master/fedcloud-ui.sh /metapipe-install
   /fedcloud-ui.sh
RUN chmod +x fedcloud-ui.sh
RUN ./fedcloud-ui.sh
RUN mkdir $HOME/.globus
RUN ln -s /metapipe-files/mmg-cluster-setup-CESNET/
   usercert.pem $HOME/.globus/usercert.pem
RUN ln -s /metapipe-files/mmg-cluster-setup-CESNET/
   userkey.pem $HOME/.globus/userkey.pem
RUN mkdir $HOME/.ssh
RUN ln -s /metapipe-files/mmg-cluster-setup-CESNET/id_rsa
    $HOME/.ssh/id_rsa
WORKDIR /metapipe-files/mmg-cluster-setup-CESNET
VOLUME ["/tmp", "/metapipe-files"]
```

Následně se MMG cluster spouští v Docker prostředí, např. pro `create.py`:

```
sudo docker run -v /home/cduongt/metapipe-docker:/
   metapipe-files -v /tmp:/tmp metapipe-docker ./create.
   py
```

8 ZÁVĚR

Cílem této diplomové práce bylo ověření přenositelnosti distribuovaných výpočtů, konkrétně služby META-pipe do cloudového prostředí, konkrétně EGI Federated Cloud.

V první kapitole byly popsány jednotlivé druhy distribuovaných výpočtů a následně byla popsána služba META-pipe, její vstupy a výstupy a její komponenty.

V další kapitole byly popsány organizace zapojené v projektech související s distribuovanými výpočty v Evropě a speficiky byly popsané ty přímo zapojené do služby META-pipe.

Ve čtvrté kapitole se práce věnovala analýze provozu META-pipe v norském prostředí NeLS, kde služba vznikla a je zde prvotní provoz. Kapitola byla zaměřená především na integraci META-pipe a nástroji, který ji spouští.

V následující kapitole bylo navrženo řešení na bázi Python skriptu obalující nástroje Terraform a Ansible, ktero bylo následně implementované a odzkoušené provozem v EGI Federated Cloud.

Nakonec se řešení částečně optimalizovalo, což bylo dosaženo pomocí úpravy původní architektury.

Přínosem této práce je ověřením toho, že služba META-pipe je portovatelná do distribuovaného prostředí cloudu a s malými úpravami je možné ji provozovat v libovolné cloudové infrastruktuře, nejen EGI Federated Cloud. Dalším z přínosů je optimalizace architektury META-pipe, čímž se zlepší využití výpočetního výkonu. Tím byl cíl práce splněn.

Literatura

[1] Espen Mikal Robertsen, Tim Kahlke, Inge Alexander Raknes, Edvard Pedersen, Erik Kjærner-Semb, Martin Ernstsen, Lars Ailo Bongo, and Nils Peder Willassen. META-pipe - Pipeline Annotation, Analysis and Visualization of Marine Metagenomic Sequence Data. *CoRR*, abs/1604.04103, 2016. URL: `http://arxiv.org/abs/1604.04103`, `arXiv:1604.04103`.

[2] Simon Andrews. FastQC. [online] Dostupné z: `https://www.bioinformatics.babraham.ac.uk/projects/fastqc/`. [cit. 2019-04-05].

[3] Katrin Leinweber. Krona. [online] Dostupné z: `https://github.com/marbl/Krona/wiki`. [cit. 2019-04-02].

[4] Johannes Goll, Douglas B Rusch, David M Tanenbaum, Mathangi Thiagarajan, Kelvin Li, Barbara Methé, and Shibu Yooseph. METAREP: JCVI Metagenomics Reports - an open source tool for high-performance comparative metagenomics. *Bioinformatics (Oxford, England)*, 26:2631–2, 10 2010. `doi:10.1093/bioinformatics/btq455`.

[5] ELIXIR Europe. EXCELERATE. [online] Dostupné z: `https://elixir-europe.org/about-us/how-funded/eu-projects/excelerate`. [cit. 2019-04-03].

[6] Luděk Matyska. EGI jako základ celoevropského výzkumného prostoru (ERA). [online] Dostupné z: `https://www.nic.cz/files/nic/doc/prezentace/IT09_Matyska.pdf`. [cit. 2019-04-03].

[7] EGI. EGI-Engage. [online] Dostupné z: `https://www.egi.eu/wp-content/uploads/2018/02/EGI-Engage_Impact_and_Results.pdf`. [cit. 2019-04-01].

[8] Enol Fernández-del Castillo, Diego Scardaci, and Álvaro López García. The EGI Federated Cloud e-Infrastructure. *ScienceDirect*, 68:196–205, 2015. `doi:http://doi.org/10.1016/j.procs.2015.09.235`.

[9] CESNET. MetaCentrum NGI. [online] Dostupné z: `https://www.metacentrum.cz/cs/`. [cit. 2019-04-03].

[10] A Agafonov, K Mattila, CD Tuan, L Tiede, IA Raknes, and LA Bongo. META-pipe cloud setup and execution [version 2; referees: 1 approved, 1 approved with reservations] . *F1000Research*, 6(2060), 2018. `doi:10.12688/f1000research.13204.2`.

[11] OCCI-WG. Open cloud computing interface – core. [online] Dostupné z: https://www.ogf.org/documents/GFD.221.pdf. [cit. 2019-04-03].

[12] Rafael Belchior. DevOps101 - First Steps on Terraform: Terraform, OpenStack, Ansible. [online] Dostupné z: https://hackernoon.com/terraform-openstack-ansible-d680ea466e22. [cit. 2019-04-06].

[13] HashiCorp. Terraform documentation. [online] Dostupné z: https://www.terraform.io/docs/index.html. [cit. 2019-03-15].

[14] CERN. CernVM-FS Documentation. [online] Dostupné z: https://cvmfs.readthedocs.io/en/stable/index.html. [cit. 2019-03-12].

[15] Wikipedia contributors. Squid (software) — Wikipedia, the free encyclopedia. [online] Dostupné z: https://en.wikipedia.org/w/index.php?title=Squid_(software)&oldid=889943789, 2019. [cit. 2019-03-18].

[16] Docker Inc. What is a container? [online] Dostupné z: https://www.docker.com/resources/what-container. [cit. 2019-03-18].

[17] R.M.L. Hochstein. *Ansible: Up and Running, 2nd Edition*. O'Reilly Media, Incorporated, 2017.

[18] Yevgeniy Brikman. *Terraform: Up and Running Writing Infrastructure As Code*. O'Reilly Media, Inc., 1st edition, 2017.

Seznam obrázků

2.1 Schéma jednotlivých komponentů a modulů META-pipe [1] 6
2.2 Program FastQC [2] 7
2.3 Zobrazení hiearchických dat v programu Krona [3] 8
2.4 Webový interface programu METAREP [4] 9
3.1 Platforma ELIXIR [5] 11
3.2 Struktura EXCELERATE projektu [5] 13
3.3 Interakce mezi EGI a NGI [6] 15
3.4 EGI-Engage v číslech [7] 16
3.5 Architektura EGI Federated Cloud [8] 17
3.6 Schéma workflow podpory uživatelů [8] 22
3.7 Mapa rozmístění hardware v rámci MetaCentra [9] 24
4.1 Integrace META-pipe v NeLS [1] 26
4.2 Integrace úložiště v Galaxy [1] 27
4.3 Integrace superpočítače a Galaxy [1] 28
4.4 Integrace LWR v superpočítači Stallo [1] 29
4.5 Architektura serveru META-pipe [10] 30
4.6 Architektura klienta META-pipe [10] 31
5.1 Umístění OCCI v rámci cloudové infrastruktury [11] 33
5.2 Architektura Terraformu [12] 35
5.3 MMG Cluster Setup - CESNET tool, Zdroj: autor 40
6.1 Závislostní graf pro vytvoření infrastruktury [13] 45
7.1 Architektura CernVM-FS [14] 54
7.2 Synchronizace CernVM-FS [14] 55
7.3 Centrální Stratum 0 repozitář CernVM-FS a replikační servery [14] . 56
7.4 Squid jako součást LAMP stacku [15] 57
7.5 Nová architektura infrastruktury MMG Cluster setup. Zdroj: autor . 59
7.6 Běžný síťový provoz master uzlu, Zdroj: autor 61
7.7 Komunikace Dockeru s Linux jádrem [16] 62
7.8 Rozdílná architektura Docker kontejnerů a tradičních virtuálních strojů [16] 63

Seznam tabulek

5.1 Porovnání Terraformu s jinými infrastructure as a code nástroji, Zdroj: autor 35
7.1 Porovnání času výpočtu před a po úpravě architektury, Zdroj: autor . 61